Logic in Action

–New Edition, November 23, 2016–

Johan van Benthem, Hans van Ditmarsch, Jan van Eijck, Jan Jaspars

Contents

Chapter 1

General Introduction

1.1 Inference, Observation, Communication

Much of our interaction with each other in daily life has to do with information processing and reasoning about knowledge and ignorance of the people around us. If I ask a simple question, like "Can you tell me where to find the Opera House?", then I convey the information that I do not know the answer, and also, that I think that you may know. Indeed, in order to pick out the right person for asking such informative questions, we need to reason about knowledge of others. It is our ability to reason in the presence of other reasoning agents that has made us historically so successful in debate, organization, and in planning collective activities. And it is reasoning in this broad sense that this course is about.

We will study informational processes of inference and information update – and while we can start dealing with these for single agents, our theories must also work interactively when many agents exchange information, say, in a conversation or a debate. As we proceed, you will see many further aspects of this program, and you will learn about mathematical models for it, some quite recent, some already very old.

Reasoning and Proof While reasoning in daily life and solving practical tasks is important, many logical phenomena become more pronounced when we look at specialized areas, where our skills have been honed to a greater degree.

To see the power of pure inference unleashed, think of mathematical proofs. Already in Greek Antiquity (and in parallel, in other cultures), logical inference provided a searchlight toward surprising new mathematical facts. In our later chapter on Proof, we will give examples, including the famous Pythagorean proof that $\sqrt{2}$ is not a rational number.The Holy Writ of this tradition are Euclid's *Elements* from around 300 BC with its formal set-up of axioms, definitions, and theorems for geometry.


EVCLID. GEOM.

PROBL. 14. PROPOS. 46.

A DATA recta linea quadratum describere.

EX PROCLO.

LIBER I. 71

SCHOLION.

THEOR. 33. PROPOS. 47.

IN rectãgulis triãgulis, quadratũ, quod a latere rectum angulum subtendente describitur, æquale est eis, quæ a lateribus rectum angulum continentibus.

(1.1)

Indeed, mathematical methods have deeply influenced the development of logic. They did so in two ways. First, mathematical proof is about the purest form of inference that exists, so it is an excellent 'laboratory' for studying inference. But also, mathematics is about the clearest way that we have for modeling phenomena and studying their properties, and logical systems of any kind, even when dealing with daily life, use mathematical techniques.

Reasoning and Observation Combinations of inference with other information sources drive the natural sciences, where experiments provide information that is just as crucial as mathematical proof. Observations about Nature made by scientists involves the same sort of information update as in simple question answering. Seeing new facts removes uncertainty. And the art is to ask the right questions, to find the right mixtures of new evidence and deduction from what we have seen already.

The same skill actually occurs in other specialized practices. Conan Doyle's famous detective Sherlock Holmes is constantly thinking about what follows from what he has seen already, but he also uses his powers of deduction to pinpoint occasions where he needs new evidence. In a famous story, the dog did not bark at night-time (and so, the intruder must have been known to the dog), but this conclusion also directs attention toward making further observations, needed to see which of the various familiar persons committed the crime.

(1.2)

Reasoning and Argumentation From crime it is only one step to lawyers and courts. Legal reasoning is another major tradition where logic is much in evidence, and we will return to this later.

1.2 The Origins of Logic

Logic as a systematic discipline dates back two and a half millennia: younger than Mathematics or the Law, but much older than most current academic disciplines, social institutions, or for that matter, religions. Aristotle and the Stoic philosophers formulated explicit systems of reasoning in Greek Antiquity around 300 BC.

(1.3)

Aristotle appearing on two Greek postal stamps The early Stoic Zeno of Citium

Independent traditions arose around that time in China and in India, which produced famous figures like the Buddhist logician Dignaga, or Gangesa, and this long tradition lives on in some philosophical schools today. Through translations of Aristotle, logic also reached the Islamic world. The work of the Persian logician Avicenna around 1000 AD was still taught in madrassa's by 1900. All these traditions have their special concerns and features, and there is a growing interest these days in bringing them closer together.

We mention this point because the cross-cultural nature of logic is a social asset beyond its scientific agenda.

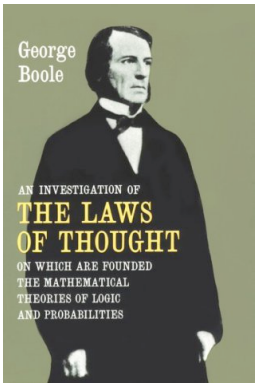
Mo Zi, founder of Mohism

Dignaga, Indian Buddhist Logician

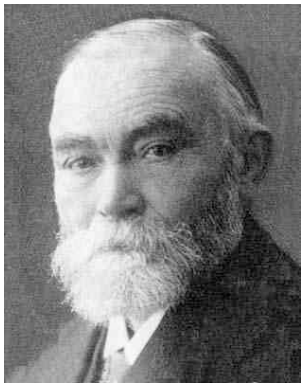
Avicenna, Persian Logician

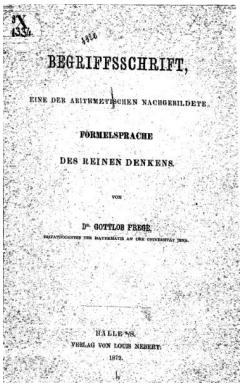
(1.4)

Still, with all due respect for this historical past that is slowly coming to light, it seems fair to say that logic made a truly major leap in the nineteenth century, and the modern logic that you will see in this course derives its basic mind-set largely from the resulting golden age of Boole, Frege, Gödel, and others: a bunch of European university professors, some quite colourful, some much less so.



George Boole on the cover of
the 'Laws of Thought' (1847),
the book that created propositional logic,
the theme of the next chapter.





Gottlob Frege with on the right
the first page of his 'Begriffsschrift' (1879),
with the system of first-order predicate logic
that can analyze much of mathematics.

(1.5)

Even so, it remains an intriguing and unsolved historical question just how and why logic arose — and we will have more to say on this below. The standard story is that great thinkers like Aristotle suddenly realized that there is structure to the human reasoning that we see all around us. Some patterns are valid and reliable, while others are not. But it has also been suggested that an interest in logic arose out of philosophical, mathematical, juridical, or even political practice. Some 'moves' worked, others did not – and people became curious to see the general reasons why.

1.3 Uses of Inference

The TV has gone dark. If it goes dark, this is due to the apparatus or the remote (or both). But the remote is working, so it must be the apparatus, and we must start repairs there. This pattern involves a logical key-word, the *disjunction* 'or':

$$A \text{ or } R, \text{ not } R. \text{ So: } A. \tag{1.6}$$

In pure form, we can also see this pattern at work in solving Sudoku puzzles. Logic also helps create new Sudoku puzzles. Start with any complete nine-digit diagram. Now pick a random slot and remove the digit in that slot. The remaining digits in the diagram still completely determine what should be in the open slot, for the digit in that slot follows by logical inference (or: by valid inference) from the other digits and the general sudoku constraints. In this way, one can go on picking filled positions at random, and checking if the digit in that position still follows from others by a valid inference. Keep doing this until no longer possible. You have now generated a minimal puzzle, and since your steps are hidden, it may take readers quite a while to figure out the unique solution.

Cognitive scientists have suggested that the primary use of logic may have been in planning. Clearly, thinking about constraints and consequences of tasks beforehand is an immense evolutionary advantage. Here is a simple illustration.

Planning a party How can we send invitations given the following constraints?

(i) John comes if Mary or Ann comes.
(ii) Ann comes if Mary does not come. (1.7)
(iii) If Ann comes, John does not.

In the chapter on propositional logic, you will learn simple techniques for solving this: for now, just try! (Here is a hint: start out with a 'maximal' invitation list *John, Ann, Mary*, and check what you have to drop to satisfy the constraints. Bear in mind that there may be several solutions to this.)

Legal reasoning We also said that daily skills can be optimized for special purposes. As we said already, inference is crucial to legal reasoning, and so is the earlier-mentioned multi-agent feature that different actors are involved: defendant, lawyer, prosecutor, judge.

The prosecutor has to prove that the defendant is guilty (G) on the basis of the available admissible evidence (E), i.e., she has to prove the conclusion G from evidence E. But the usual 'presumption of innocence' means that the lawyer has another logical task: viz. making it plausible that G does not follow from E. This does not require her to demonstrate that her client is innocent: she just needs to paint one scenario consistent with the evidence E where G fails, whether it is the actual one or not.

Logical key-words There are certain logical key-words driving patterns of inference. Expressions like 'not', 'and', 'or', 'if then' are sentence forming constructions that classify situations as a whole. What we mean by this is that these expressions can be used to construct new sentences from existing sentences. From "it is raining" to "it is not raining". From "it is raining" and "it is wet" to "if it is raining then it is wet", and so on.

But there are other expressions that tell us more about the internal structure of these situations, in terms of objects and their properties and relations. "Hans is friendly" ascribes a property to a person. "Hans and Jan are colleagues" describes a relation between two persons. Historically, the most important example are quantifiers, expressions of quantity such as 'all', 'every', 'some' or 'no'. "All logicians are friendly" describes how the properties of being a logician and being friendly are related, using the quantifier 'all'.

The view of inference as the result of replacing some parts in expressions by variable parts, so that only logical key-words and variables remain, can already be found in the work of the Bohemian philosopher and priest Bernhard Bolzano (1781 – 1848).

(1.8)

Bernard Bolzano

Aristotle's syllogisms listed the basic inference patterns with quantifiers, such as

All humans are animals, no animals are mortal. So, no humans are mortal. (1.9)

This is a valid inference. But the following is not valid:

Not all humans are animals, no animals are mortal. So, some humans are mortal. (1.10)

Syllogistic forms were long considered the essence of logical reasoning, and their format has been very influential until the 19th century. Today, they are still popular test cases for psychological experiments about human reasoning.

Quantifiers are essential to understanding both ordinary and scientific discourse. If you unpack standard mathematical assertions, you will find any amount of stacked quantifiers. For instance, think of saying that 7 is a *prime number*. This involves:

All of 7's divisors are either equal to 1 or to 7, (1.11)
where x divides y if *for some* z: $x \cdot z = y$.

Here 'all of' and 'for some' are the quantifiers that provide the logical glue of the explanation of what it means to be prime, or to be a divisor. Other examples with many quantifiers occur in Euclid's geometry and spatial reasoning in general.

We will devote two entire chapters to the logic of the quantifiers 'all', 'some', given its central importance. Actually, natural language has many further quantifier expressions, such as 'three', 'most', 'few', 'almost all', or 'enough'. This broad repertoire raises many issues of its own about the expressive and communicative function of logic, but we sidestep these here.

Many further logical key-words will emerge further on in this course, including expressions for reasoning about knowledge and action.

Another crucial feature of logic, that makes it a true scientific endeavour in a systematic sense, is the turning of human reasoning to itself as a subject of investigation. But things go even one step further. Logicians study reasoning practices by developing mathematical models for them – but then, they also make these systems themselves into a new object of investigation.

Logical systems Indeed, Aristotle already formulated explicit logical systems of inference in his Syllogistics, giving all valid rules for syllogistic quantifier patterns. Interestingly, Aristotle also started the study of grammar, language looking at language — and earlier than him, the famous Sanskrit grammarian Panini had used mathematical systems there, creating a system that is still highly sophisticated by modern standards:

(1.12)

This mathematical system building tradition has flourished over time, largely (but not exclusively) in the West. In the nineteenth century, George Boole gave a complete analysis of propositional logic for reasoning with sentential operators like 'not', 'and', 'or', that has become famous as the 'Boolean algebra' that underlies the switching circuits of your computer. Boole showed that all valid principles of propositional reasoning can be derived from a simple calculus, by purely algebraic manipulations. We will explain how this works later on in this course.

Subsequently, Frege gave formal systems for reasoning with quantifiers in ways that go far beyond Aristotle's Syllogistic. Over time, systems in this line have proved strong enough to formalize most of mathematics, including its foundational *set theory*.

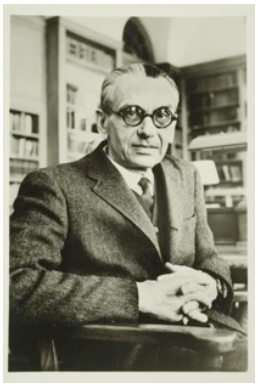
Foundations of mathematics Through this process of scrutiny, mathematical and logical theories themselves become objects of investigation. And then, some startling discoveries were made. For instance, here is the so-called Russell Paradox from the foundations of set theory.

Set theory is a general way of talking about collections of entities What the Russell paradox tells us is that we have to be very careful in how to express ourselves in talking about collections of entities. For suppose anything goes in defining sets, so that, if we have a description we can construct the set of all things satisfying the description. Then the following can happen.

> Some sets contain themselves as a member (e.g., the set of all non-teaspoons is not a teaspoon, so the set of non-teaspoon has itself as a member). Others do not (for instance, the set of all teaspoons is not itself a teaspoon.) Now consider the set R of all sets that do not have themselves as members. It is easy to see that R is a member of R if and only if R is *not* a member of R: and that is a contradiction.

The sort of reasoning that leads to this paradox will be taken up in several later chapters. The formal definition of the Russell set R is: $R = \{x \mid x \notin x\}$. The paradoxical statement is: $R \in R$ if and only if $R \notin R$. If you have never seen the symbol $\in$ or the bracket notation $\{x \mid \ldots\}$ then you should at some point consult Appendix A to catch up with the rest of us.

The foundational problems in the development of logic illustrated by Russell's paradox led to the so-called foundational study of mathematics, which investigates formal properties of mathematical theories, and power and limits of proofs. A famous name here is Kurt Gödel, probably the greatest figure in the history of logic. His incompleteness theorems are fundamental insights into the scope and reliability of mathematics, that got him on the TIME 2001 list of most influential intellectuals of the twentieth century. But in Amsterdam, we also cite our own L.E.J. Brouwer, the father of 'intuitionistic logic', an important program in the foundations of mathematics and computation. These mathematical theoretical aspects of logic belong more properly to an advanced course, but we will give you some feeling for this theme further on in this book.



Kurt Gödel

Brouwer on a Dutch post stamp

(1.13)

1.4 Logic and Other Disciplines

Looking at the list of topics discussed above, you have seen switches from language and conversation to mathematics and computation. Indeed, in a modern university, logic lies at a cross-roads of many academic disciplines. This course will make you acqainted with a number of important *systems* for doing logic, but it will also draw many connections between logic and related disciplines. We have already given you a taste of what logic has to do with *mathematics*. Mathematics supplies logic with its techniques, but conversely, logic can also be used to analyze the foundations of mathematics. Now we look at a few more important alliances.

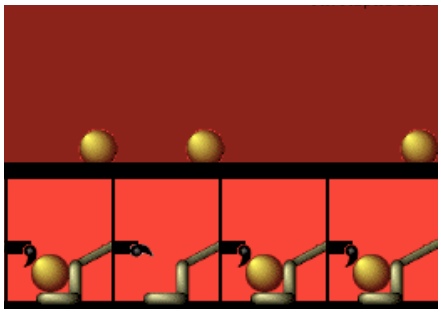
Logic, language and philosophy Perhaps the oldest connection of logic is with *philosophy*. Logic has to do with the nature of assertions, meaning, and knowledge, and philosophers have been interested in these topics from the birth of philosophy. Logic can serve as a tool for analyzing philosophical arguments, but it is also used to *create* philosophical systems. Logical forms and calculating with these is a role model for conceptual abstraction. It has even been claimed that logical patterns of the sort sketched here are close to being a 'universal language of thought'.

But it will also be clear that logic has much to do with *linguistics*, since logical patterns arise from abstraction out of the grammar of ordinary language, and indeed, logic and linguistics share a long history from Antiquity through the Middle Ages.

Logic and computation Another long-standing historical theme interleaves logic and *computation*. Since the Middle Ages, people have been fascinated by machines that would

automate reasoning, and around 1700, Leibniz

Gottfried Wilhelm von Leibniz

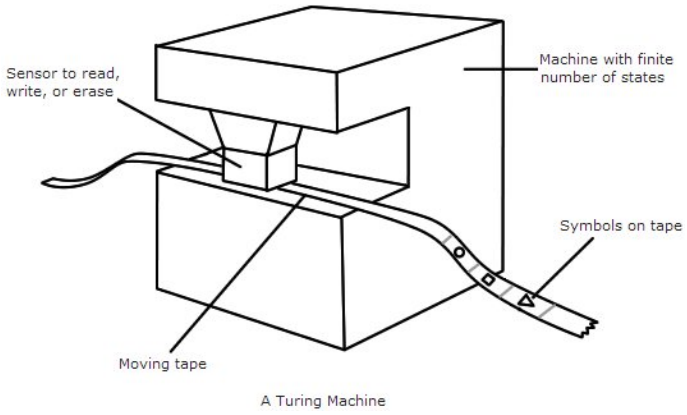


The first binary addition mechanism as described by Leibniz in a paper called 'Mechanica Dyadica' (around 1700)

(1.14)

realized that logical inference may be viewed as a sort of computation, though not with ordinary but with binary numbers. A straight line runs from here to modern computers and computer science, and the seminal work of Turing and others.

Alan Turing



A 'Turing Machine'

(1.15)

Logic and games While mathematics, philosophy, linguistics, and computer science are old neighbours of logic, new interfaces keep emerging. We end with one directed toward the social and behavioural sciences. As we have said before, logic had its origins in a tradition of conversation, debate, and perhaps legal procedure. This brings us back to our earlier theme that much logical behaviour is interactive, crucially involving other persons.

Argumentation itself is a key example. There are different parties playing different roles, and reacting to each other over time. This clearly has the structure of a game. In such a game logical operations like 'or', 'and' and 'not' function as a sort of 'switches', not just in a Boolean computer, but also in discussion. When I defend that 'A or B', then you can hold me to this, and I have to choose eventually which of the two I will defend. Thus, a disjunction offers a choice to its defender — and likewise, a conjunction 'A and B'

offers a choice to the attacker: since the defender is committed to both parts. Interesting interactions also arise by means of the third item of Boolean algebra: logical negation. This triggers a role switch: defending 'not A' is attacking 'A', and vice versa. Indeed, being able to 'put yourself in another person's place' has been called the quintessential human cognitive achievement.

In this way, logic comes to describe the structure of rational interaction between conversation partners. Traditions of vigorous regimented logical debating games flourished in the Middle Ages, and they still do in some parts of the world:

(1.16)

Karma Guncho, ten monasteries battle each other
on Buddhist philosophy using logical analysis.

In this game setting, we may call an inference valid if the defender of the conclusion has a 'winning strategy': that is, a rule for playing which will always lead her to win the game against any defender of the premises, whatever that person brings up over time.

But if logic has much to do with games, then it also has links with economic game theory, and not surprisingly, this is another flourishing interface today. We will develop this topic in greater depth in a separate chapter, but now you know why.

1.5 Overview of the Course

In this course, logic will be presented as a key element in the general study of reasoning, information flow and communication: topics with a wide theoretical and practical reach. The course starts with introductions to three important systems of reasoning: propositional logic (Chapter 2), syllogistics (Chapter 3), and predicate logic (Chapter 4). Together, these describe situations consisting of objects with a great variety of structure, and in doing so, they cover many basic patterns that are used from natural language to the depths of mathematics.

Next, we move on to the newer challenges on a general agenda of studying information flow. The first is agents having information and interacting through questions, answers, and other forms of communication. This social aspect is crucial if you think about how we use language, or how we behave in scientific investigation. We will model observation and reasoning in a multi-agent setting, introducing the logic of knowledge in Chapter **??**.

To model the dynamic aspect of all this, we turn to the basic logic of action in Chapter 6. And Chapter 7 takes up a more recent theme: the use of games as a model of interaction. These bring together many of the separate topics in the course so far.

The next group of chapters then develop three logical methods more in detail. Chapter 8 is about precise ways of testing logical validity, that give you a sense of how a significant logical calculus really works. Chapter 9 goes into mathematical proof and its structures. Chapter 10 gives more details on the many relations between logic and computation.

In all of these chapters, and even more in the internet version of this text, you will find links to topics in philosophy, mathematics, linguistics, cognition and computation, and you will discover that logic is a natural 'match-maker' between these disciplines.

We have tried to give an indication of the difficulty of the exercises, as follows: ♡ indicates that a problem is easy (solving the problems marked as ♡ can be used as a test to check that you have digested the explanations in the text), ♠ indicates that a problem is a bit harder than average, and ♠♠ indicates that a problem is quite hard. If you feel you can handle an extra challenge, you are encouraged to try your hand at these.

Classical Systems

Chapter 2

Propositional Logic

Overview The most basic logical inferences are about combinations of sentences, expressed by such frequent expressions as 'not', 'and', 'or', 'if, then'. Such combinations allow you to describe situations, and what properties these situations have or lack: something is 'not this, but that'. You could call this reasoning about 'classification', and it is the basis of any description of the world. At the same time, these logical sentence combinations are also fundamental in another sense, as they structure how we communicate and engage in argumentation. When you disagree with a claim that someone makes, you often try to derive a consequence ('if then') whose negation ('not') is easier to show. We will study all these patterns of reasoning below.

More precisely, in this first chapter you will be introduced to *propositional logic*, the logical system behind the reasoning with 'not', 'and', 'or', 'if, then' and other basic sentence-combining operators. You will get acquainted with the notions of formula, logical connective, truth, valid consequence, information update, formal proof, and expressive power, while we also present some backgrounds in computation and cognition.

2.1 Reasoning in Daily Life

Logic can be seen in action all around us:

> In a restaurant, your Father has ordered Fish, your Mother ordered Vegetarian, and you ordered Meat. Out of the kitchen comes some new person carrying the three plates. What will happen?

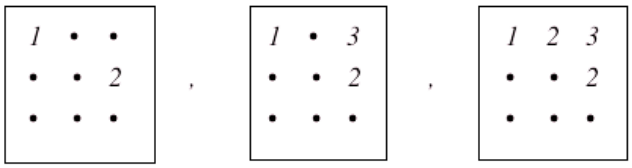
We have know this from experience. The waiter asks a first question, say "Who ordered the meat?", and puts that plate. Then he asks a second question "Who has the fish?", and puts that plate. And then, without asking further, he knows he has to put the remaining plate in front of your Mother. What has happened here?

Starting at the end, when the waiter puts the third plate without asking, you see a major logical act 'in broad daylight': the waiter draws a conclusion. The information in the two answers received allows the waiter to infer automatically where the third dish must go. We represent this in an *inference schema* with some special notation (F for "fish", M for "meat", V for "vegetarian"):

$$F \text{ or } V \text{ or } M, \text{not } M, \text{ not } F \Longrightarrow V. \tag{2.1}$$

This formal view has many benefits: one schema stands for a wide range of inferences, for it does not matter what we put for F, V and M.

Inferences often come to the surface especially vividly in puzzles, where we exercise our logical abilities just for the fun of it. Think of successive stages in the solution of a 3×3 Sudoku puzzle, produced by applying the two basic rules that each of the 9 positions must have a digit, but no digit occurs twice on a row or column:

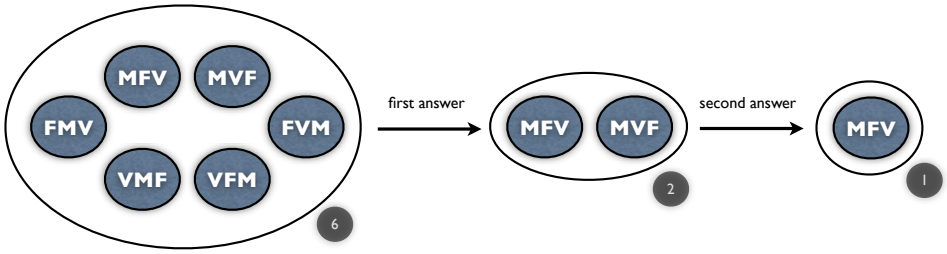


(2.2)

Each successive diagram displays a bit more explicit information about the solution, which is already implicitly determined by the initial placement of the two digits 1, 2. And the driving mechanism for these steps is exactly our Restaurant inference. Think of the step from the first to the second picture. The top right dot is either 1, 2 or 3. It is not 1. It is not 2. Therefore, it has to be 3.

But is much more information flow in this Restaurant scene. Before his final inference, the waiter first actively sought to find out enough facts by another typical information-producing act, viz. asking a question. And the answers to his two questions were also crucial.

The essence of this second process is a form of computation on information states. During a conversation, information states of people – singly, and in groups – change over time, triggered by communicative events. The Restaurant scenario starts with an initial information state consisting of six options, all the ways in which three plates can be distributed over three people $(MFV, MVF, ...)$. The answer to the first question then reduces this to two (the remaining orders FV, VF), and the answer to the second question reduces this to one, zooming in on just the actual situation (for convenience, assume it is MFV). This may be pictured as a diagram ('video') of successive updates:



(2.3)

2.2 Inference Patterns, Validity, and Invalidity

Consider the following statement from your doctor:

> If you take my medication, you will get better.
> But you are not taking my medication.
> ---
> So, you will not get better.

(2.4)

Here the word 'so' (or 'therefore', 'thus', etc.) suggests the drawing of a conclusion from two pieces of information: traditionally called the 'premises'. We call this an act of inference. Now, as it happens, this particular inference is not compelling. The conclusion might be false even though the two premises are true. You might get better by taking that greatest medicine of all (but so hard to swallow for modern people): just wait. Relying on a pattern like this might even be pretty dangerous in some scenarios:

> If I resist, the enemy will kill me.
> But I am not resisting.
> ---
> So, the enemy will not kill me.

(2.5)

Now contrast this with another pattern:

> If you take my medication, you will get better.
> But you are not getting better.
> ---
> So, you have not taken my medication.

(2.6)

This is valid: there is no way that the two stated premises can be true while the conclusion is false. It is time for a definition. Broadly speaking,

> we call an inference *valid* if there is 'transmission of truth': in every situation where all the premises are true, the conclusion is also true.

Stated differently but equivalently, an inference is valid if it has no 'counter-examples': that is, situations where the premises are all true while the conclusion is false. This is a crucial notion to understand, so we dwell on it a bit longer.

What validity really tells us While this definition makes intuitive sense, it is good to realize that it may be weaker than it looks a first sight. For instance, a valid inference with two premises

$$P_1, P_2, \text{ so } C \tag{2.7}$$

allows many combinations of truth and falsity. If any premise is false, nothing follows about the conclusion. In particular, in the second doctor example, the rule may hold (the first premise is true), but you are getting better (false second premise), and you did take the medication (false conclusion). Of all eight true-false combinations for three sentences, validity rules out 1 (true-true-false)! The most you can say for sure thanks to the validity can be stated in one of two ways:

$$\begin{array}{l}\text{(a) if all premises are true, then the conclusion is true} \\ \text{(b) if the conclusion is false, then at least one premise is false}\end{array} \tag{2.8}$$

The first version is how people often think of logic: adding more things that you have to accept given what you have accepted already. But there is an equally important use of logic in *refuting* assertions, perhaps made by your opponents. You show that some false consequence follows, and then cast doubt on the original assertion. The second formulation says exactly how this works. Logical inferences also help us see what things are false — or maybe more satisfyingly, refute someone else. But note the subtlety: a false conclusion does not mean that all premises were false, just that at least one is. Detecting this bad apple in a basket may still take further effort.

To help you understand both aspects of validity, consider the tree below: representing a 'complex argument' consisting of individual inferences with capital letters for sentences, premises above the bar, and the conclusion below it. Each inference in the tree is valid:

$$\dfrac{\dfrac{\dfrac{A \quad B \quad C}{A \qquad D}}{E} \qquad \dfrac{E}{F} \qquad \dfrac{A}{B}}{G} \tag{2.9}$$

You are told reliably that sentence A is true and G is false. For which further sentences that occur in the tree can you now determine their truth and falsity? (The answer is that A, B, are true, C, D, E , G are false, while we cannot tell whether F is true or false.)

Inference patterns The next step in the birth of logic was the insight that the validity and invalidity here have to do with abstract patterns, the shapes of the inferences, rather than their specific content. Clearly, the valid second argument would also be valid in the following concrete form, far removed from doctors and medicine:

$$\frac{\begin{array}{l}\text{If the enemy cuts the dikes, Holland will be inundated.}\\ \text{Holland is not inundated.}\end{array}}{\text{So, the enemy has not cut the dikes.}} \qquad (2.10)$$

This form has variable parts (we have replaced some sentences by others), but there are also constant parts, whose meaning must stay the same, if the inference is to be valid. For instance, if we also replace the negative word 'not' by the positive word 'indeed', then we get the clearly invalid inference:

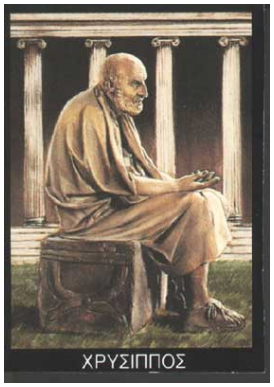
$$\frac{\begin{array}{l}\text{If the enemy cuts the dikes, Holland will be inundated.}\\ \text{Holland is indeed inundated.}\end{array}}{\text{So, the enemy has indeed cut the dikes.}} \qquad (2.11)$$

For counter-examples: the inundation may be due to faulty water management, rain, etc.

To bring out the relevant shared underlying form of inferences, we need a notation for both fixed and variable parts. We do this by using variable letters for expressions that can be replaced by others in their linguistic category, plus special notation for key expressions that determine the inference, often called the logical constants.

2.3 Classification, Consequence, and Update

Classification The main ideas of propositional logic go back to Antiquity (the Stoic philosopher Chrysippus of Soli, c.280–c.207 BC), but its modern version starts in the nineteenth century, with the work of the British mathematician George Boole (1815–1864).



Chrysippus

George Boole

Our earlier examples were essentially about combinations of propositions (assertions expressed by whole sentences). From now on, we will indicate basic propositions by letters p, q, etcetera. A finite number of such propositions generates a finite set of possibilities, depending on which are true and which are false. For instance, with just p, q there are four true/false combinations, that we can write as

$$pq, p\overline{q}, \overline{p}q, \overline{pq} \tag{2.12}$$

where p represents that p is true and $\overline{p}$ that p is false. Thus, we are interested in a basic logic of this sort of classification. (Note that $\overline{p}$ is not meant as a logical proposition here, so that it is different from the negation not-p that occurs in inferences that we will use below. The distinction will only become clear later.)

Drawing consequences Now consider our earlier examples of valid and invalid arguments. For instance,

(a) the argument "from if-p-then-q and not-p to not-q" was invalid,

whereas

(b) the argument "from if-p-then-q, not-q to not-p" was valid.

Our earlier explanation of validity for a logical consequence can now be sharpened up. In this setting, it essentially says the following:

> each of the above four combinations that makes the premises true must also make the conclusion true.

You can check whether this holds by considering all cases in the relevant list that satisfy the premises. For instance, in the first case mentioned above,

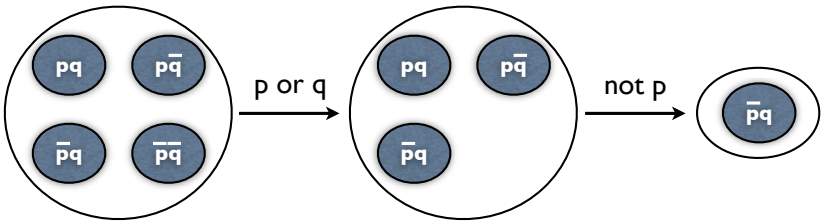
(a) not-p is true at $\overline{p}q$ and $\overline{pq}$. if-p-then-q holds also in these two situations, since the condition p is not true. So, the first of the two situations, $\overline{p}q$, support the two premises but the conclusion not-q is false in this situation. The argument is therefore invalid!

For the second case we get

(b) not-q is true at $p\overline{q}$ and $\overline{pq}$. while if-p-then-q only holds in the second, so $\overline{pq}$ is the only situation in which all the premises hold. In this situation not-p also holds, and therefore, the argument is valid.

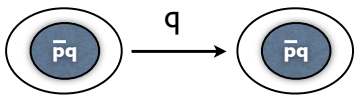
Updating information Propositional logic describes valid (and invalid) inference patterns, but it also has other important uses. In particular, it describes the information flow in earlier examples, that may arise from observation, or just facts that are being told.

With the set of all combinations present, we have no information about the actual situation. But we may get additional information, ruling out options. To see how, consider a simple party, with just two possible invitees Mary and John. We write p and q, respectively, for "Mary comes to the party" and "John comes to the party". Suppose that we are first told that at least one of the invitees comes to the party: p-or-q. Out of four possible situations this new information rules out just one, viz. $\overline{pq}$. Next, the we learn that not-p. This rules out two more options, and we are left with only the actual situation $\overline{p}q$. Here is a 'video-clip' of the successive information states, that get 'updated' by new information:



(2.13)

Incidentally, you can now also see why the conclusion q is a valid inference from 'p or q' and 'not p'. Adding the information that q does not change the final information state, nothing is ruled out:



(2.14)

But if adding the information that q does not change anything, this means that q is already true. So the truth of q is guaranteed by the fact that the two earlier updates have taken place. This must mean that q is logically implied by the earlier formulas.

Exercise 2.1 Consider the case where there are three facts that you are interested in. You wake up, you open your eyes, and you ask yourself three things: "Have I overslept?", "Is it raining?",

"Are there traffic jams on the road to work?". To find out about the first question, you have to check your alarm clock, to find out about the second you have to look out of the window, and to find out about the third you have to listen to the traffic info on the radio. We can represent these possible facts with three basic propositions, p, q and r, with p expressing "I have overslept", q expressing "It is raining", and r expressing "There are traffic jams." Suppose you know nothing yet about the truth of your three facts. What is the space of possibilities?

Exercise 2.2 (Continued from previous exercise.) Now you check your alarm clock, and find out that you have not overslept. What happens to your space of possibilities?

Toward a system Once we have a system in place for these tasks, we can do many further things. For instance, instead of asking whether a given inference is valid, we can also just look at given premises, and ask what would be a most informative conclusion. Here is a case that you can think about (it is used as a basic inference step to program computers that perform reasoning automatically):

Exercise 2.3 You are given the information that p-or-q and (not-p)-or-r. What can you conclude about q and r? What is the strongest valid conclusion you can draw? (A statement is stronger than another statement if it rules out more possibilities.)

A precise system for the above tasks can also be *automated*, and indeed, propositional logic is historically important also for its links with computation and computers. Computers become essential with complex reasoning tasks, that require many steps of inference or update of the above simple kinds, and logical systems are close to automated deduction. But as we shall see later in Section 2.10, there is even a sense in which propositional logic *is* the language of computation, and it is tied up with deep open problems about the nature of computational complexity.

But the start of our story is not in computation, but in natural language. We will identify the basic expressions that we need, and then sharpen them up in a precise notation.

2.4 The Language of Propositional Logic

Reasoning about situations involves complex sentences with the 'logical connectives' of natural language, such as 'not', 'and', 'or' and 'if .. then'. These are not the only expressions that drive logical reasoning, but they do form the most basic level. We could stay close to natural language itself to define our system (traditional logicians often did), but it has become clear over time that working with well-chosen notation makes things much clearer, and easier to manipulate. So, just like mathematicians, logicians use formal notations to improve understanding and facilitate computation.

From natural language to logical notation As we have seen in Section 2.3, logical forms lie behind the valid inferences that we see around us in natural language. So we need a good notation to bring them out. For a start, we will use special symbols for the key logical operator words:

Symbol	In natural language	Technical name
$\neg$	not	negation
$\wedge$	and	conjunction
$\vee$	or	disjunction
$\rightarrow$	if ... then	implication
$\leftrightarrow$	if and only if	equivalence

(2.15)

Other notations occur in the literature, too: some dialects have & for $\wedge$, and $\equiv$ for $\leftrightarrow$. We write small letters for basic propositions p, q, etcetera. For arbitrary propositions, which may contain connectives as given in the table (2.15), we write small Greek letters φ, ψ, χ, etc.

Inclusive and exclusive disjunction The symbol $\vee$ is for inclusive disjunction, as in 'in order to pass the exam, question 3 or question 4 must have been answered correctly'. Clearly, you don't want to be penalized if both are correct! This is different from the *exclusive disjunction* (most often written as $\oplus$), as in 'you can marry Snowwhite or Cinderella'. This is not an invitation to marry both at the same time. When we use the word 'disjunction' without further addition we mean the inclusive disjunction.

Now we can write logical forms for given assertions, as 'formulas' with these symbols. Consider a card player describing the hand of her opponent:

Sentence	"He has an Ace if he does not have a Knight or a Spade"
Logical formula	$\neg(k \vee s) \rightarrow a$

It is useful to see this process of formalization as something that is performed in separate steps, for example, as follows. In cases where you are in doubt about the formalization of a phrase in natural language, you can always decide to 'slow down' to such a stepwise analysis, to find out where the crucial formalization decision is made.

He has an Ace if he does not have a Knight or a Spade,
if (he does not have a Knight or a Spade), then (he has an Ace),
(he does not have a Knight or a Spade) $\rightarrow$ (he has an Ace),
not (he has a Knight or a Spade) $\rightarrow$ (he has an Ace),
$\neg$ (he has a Knight or a Spade) $\rightarrow$ (he has an Ace),
$\neg$ ((he has a Knight) or (he has a Spade)) $\rightarrow$ (he has an Ace),
$\neg$ ((he has a Knight) $\vee$ (he has a Spade)) $\rightarrow$ (he has an Ace),
$\neg(k \vee s) \rightarrow a$

In practice, one often also sees mixed notations where parts of sentences are kept intact, with just logical keywords in formal notation. This is like standard mathematical language, that mixes symbols with natural language. While this mixing can be very useful (the notation enriches the natural language, and may then be easier to absorb in cognitive practice), you will learn more here by looking at the extreme case where the whole sentence is replaced by a logical form.

Ambiguity The above process of taking natural language to logical forms is not a routine matter. There can be quite some slack, with genuine issues of interpretation. In particular, natural language sentences can be *ambiguous*, having different interpretations. For instance, another possible logical form for the card player's assertion is the formula

$$(\neg k \vee s) \rightarrow a \tag{2.16}$$

Check for yourself that this says something different from the above. One virtue of logical notation is that we see such differences at a glance: in this case, by the placement of the brackets, which are auxiliary devices that do not occur as such in natural language (though it has been claimed that some actual forms of expression do have 'bracketing functions').

Sometimes, the logical form of what is stated is even controversial. According to some people, 'You will get a slap (s), unless you stop whining ($\neg w$)' expresses the implication $w \rightarrow s$. According to others, it expresses the equivalence $w \leftrightarrow s$. Especially, negations in natural language may quickly get hard to grasp. Here is a famous test question in a psychological experiment that many people have difficulty with. How many negations are there, and what does the stacking of negations mean in the following sentence:

> "Nothing is too trivial to be ignored?"

Formal language and syntactic trees Logicians think of the preceding notations, not just as a device that can be inserted to make natural language more precise, but as something that is important on its own, namely, an artificial or formal language.

You can think of formulas in such a language as being constructed, starting from basic propositions, often indicated by letters p, q, etcetera, and then applying logical operations, with brackets added to secure unambiguous readability.

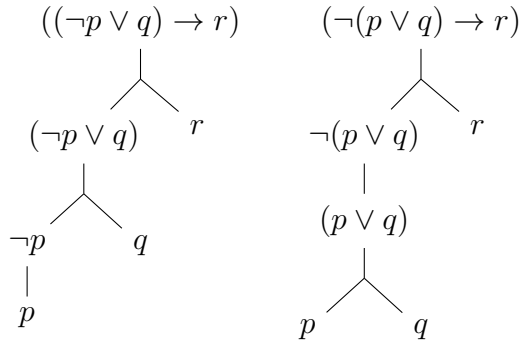
Example 2.4 The formula $((\neg p \vee q) \rightarrow r)$ is created stepwise from proposition letters p, q, r by applying the following construction rules successively:

(a) from p, create $\neg p$,

(b) from $\neg p$ and q, create $(\neg p \vee q)$

(c) from $(\neg p \vee q)$ and r, create $((\neg p \vee q) \rightarrow r)$

This construction may be visualized in *trees* that are completely unambiguous. Here are trees for the preceding example plus a variant that we already noted above. Mathematically, bracket notation and tree notation are equivalent — but their cognitive appeal differs, and trees are widely popular in mathematics, linguistics, and elsewhere:



This example has prepared us for the formal presentation of the language of propositional logic. There are two ways to go about this, they amount to the same: an 'inductive definition' (for this technical notion, see Appendix A). Here is one way:

> Every proposition letter ($p, q, r, \ldots$) is a formula. If φ is a formula, then $\neg\varphi$ is also a formula. If φ_1 and φ_2 are formulas, then $(\varphi_1 \wedge \varphi_2)$, $(\varphi_1 \vee \varphi_2)$, $(\varphi_1 \rightarrow \varphi_2)$ and $(\varphi_1 \leftrightarrow \varphi_2)$ are also formulas. Nothing else is a formula.

We can now clearly recognize that the way we have constructed the formula in the example above is exactly according to this pattern. That is merely a particular instance of the above definition. The definition is formulated in more abstract terms, using the formula variables φ_1 and φ_2. An even more abstract specification, but one that amounts to exactly the same inductive definition, is the so-called BNF specification of the language of propositional logic. BNF stands for 'Backus Naur Form', after the computer scientists John Backus and Peter Naur who introduced this device for the syntax of programming languages.

Definition 2.5 (Language of propositional logic) Let P be a set of proposition letters and let $p \in P$.

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

We should read such a definition as follows. In the definition we define objects of the type 'formula in propositional logic', in short: formulas. The definition starts by stating that every atomic proposition is of that type, i.e., is a formula. Then it says that if an object is of type φ, then $\neg\varphi$ is also of type φ. Note that it does not say that $\neg\varphi$ is the same formula φ. It merely says that both can be called 'formula'. This definition then helps us to construct concrete formulas step by step, as in the previous example.

Backus Naur form is an example of linguistic specification. In fact, BNF is a computer science re-invention of a way to specify languages that was proposed in 1956 by the linguist Noam Chomsky.

In practice we often do not write the parentheses, and we only keep them if their removal would make the expression ambiguous, as in $p \vee q \wedge r$. This can mean $((p \vee q) \wedge r)$ but also $(p \vee (q \wedge r))$ and that makes quite a difference. The latter is already true if only p is true, but the former requires r to be true. Or take a natural language example: "Haddock stays sober or he drinks and he gets angry."

Exercise 2.6 Write in propositional logic:

- I will only go to school if I get a cookie now.
- John and Mary are running.
- A foreign national is entitled to social security if he has legal employment or if he has had such less than three years ago, unless he is currently also employed abroad.

Exercise 2.7 Which of the following are formulas in propositional logic:

- $p \rightarrow \neg q$
- $\neg\neg \wedge q \vee p$
- $p\neg q$

Exercise 2.8 Construct trees for the following formulas:

- $(p \wedge q) \rightarrow \neg q$
- $q \wedge r \wedge s \wedge t$ (draw all possible trees: think of bracket arrangements).

Exercise 2.9 From the fact that several trees are possible for $q\wedge r\wedge s\wedge t$, we see that this expression can be read in more than one way. Is this ambiguity harmful or not? Why (not)? If you find this hard to answer, think of a natural language example.

A crucial notion: pure syntax Formulas and trees are pure symbolic forms, living at the level of syntax, as yet without concrete meaning. Historically, identifying this separate level of form has been a major abstraction step, that only became fully clear in 19th century mathematics. Most uses of natural language sentences and actual reasoning come with meanings attached, unless very late at parties. Pure syntax has become the basis for many connections between logic, mathematics, and computer science, where purely symbolic processes play an important role.

Logic, language, computation, and thought The above pictures may remind you of parse trees in grammars for natural languages. Indeed, translations between logical forms and linguistic forms are a key topic at the interface of logic and linguistics, which has also started working extensively with mathematical forms in the 20th century. Connections between logical languages and natural language have become important in Computational Linguistics and Artificial Intelligence, for instance when interfacing humans with computers and symbolic computer languages. In fact, you can view our syntax trees in two ways, corresponding to two major tasks in these areas. 'Top down' they analyze complex expressions into progressively simpler ones: a process of *parsing* given sentences. But 'bottom up' they construct new sentences, a task called language generation.

But also philosophically, the relation between natural and artificial languages has been long under debate. The more abstract level of logical form has been considered more 'universal' as a sort of 'language of thought', that transcends differences between natural languages (and perhaps even between cultures). You can also cast the relation as a case of replacement of messy ambiguous natural language forms by clean logical forms for reasoning and perhaps other purposes — which is what the founding fathers of modern logic had in mind, who claimed that natural languages are 'systematically misleading'. But less radically, and perhaps more realistic from an empirical cognitive viewpoint, you can also see the relation as a way of creating *hybrids* of existing and newly designed forms of expression. Compare the way the language of mathematicians consists of natural language plus a growing fund of notations, or the way in which computer science extends our natural repertoire of expression and communication.

2.5 Semantic Situations, Truth Tables, Binary Arithmetic

Differences in formal syntax often correspond to differences in meaning: the above two trees are an example. To explain this in more detail, we now need a semantics that, for a start, relates syntactic objects like formulas to truth and falsity in semantic situations. Thus, formulas acquire meaning in specific settings, and differences in meaning between formulas are often signalled by differences in truth in some situation.

Truth values and valuations for atoms As we said already, each set of proposition letters $p, q, r, \ldots$ generates a set of different *situations*, different ways the actual world might be, or different states that it could be in (all these interpretations make sense in applications). Three proposition letters generate $2^3 = 8$ situations:

$$\{pqr, pq\overline{r}, p\overline{q}r, p\overline{qr}, \overline{p}qr, \overline{p}q\overline{r}, \overline{pq}r, \overline{pqr}\} \tag{2.17}$$

Here proposition letters stand for 'atomic propositions', while logical operations form 'molecules'. Of course this is just a manner of speaking, since what counts as 'atomic' in a given application is usually just our decision 'not to look any further inside' the proposition. A convenient mathematical view of situations is as functions from atomic propositions to truth values 1 ('true') and 0 ('false'). For instance, the above situation $p\overline{q}r$ corresponds to the function sending p to 1, q to 0, and r to 1. An alternative notation for truth values is t and f, but we use numbers for their suggestive analogy with binary arithmetic (the heart of computers). We call these functions V *valuations*; $V(\varphi) = 1$ says that the formula φ is true in the situation (represented by) V, and $V(\varphi) = 0$ says that the formula φ is false in the situation V. For $V(\varphi) = 1$ we also write $V \models \varphi$ and for $V(\varphi) = 0$ we also write $V \not\models \varphi$. One can read $V \models \varphi$ as "V makes true φ", or as "V *satisfies* φ" or "V is a *model* of φ". The notation using $\models$ will reappear in later chapters.

Boolean operations on truth values Any complex sentence constructed from the relevant atomic proposition letters is either true or false in each situation. To see how this works, we first need an account for the meaning of the logical operations. This is achieved by assigning them Boolean operations on the numbers 0, 1, in a way that respects (as far as reasonable) their intuitive usage. For instance, if $V(\varphi) = 0$, then $V(\neg\varphi) = 1$, and vice versa; and if $V(\varphi) = 1$, then $V(\neg\varphi) = 0$, and vice versa. Such relations are easier formatted in a table.

Definition 2.10 (Semantics of propositional logic) A valuation V is a function from proposition letters to truth values 0 and 1. The value or meaning of complex sentences is computed from the value of basic propositions according to the following truth tables.

φ	$\neg\varphi$
0	**1**
1	**0**

φ	ψ	$\varphi \wedge \psi$	$\varphi \vee \psi$	$\varphi \rightarrow \psi$	$\varphi \leftrightarrow \psi$
0	0	**0**	**0**	**1**	**1**
0	1	**0**	**1**	**1**	**0**
1	0	**0**	**1**	**0**	**0**
1	1	**1**	**1**	**1**	**1**

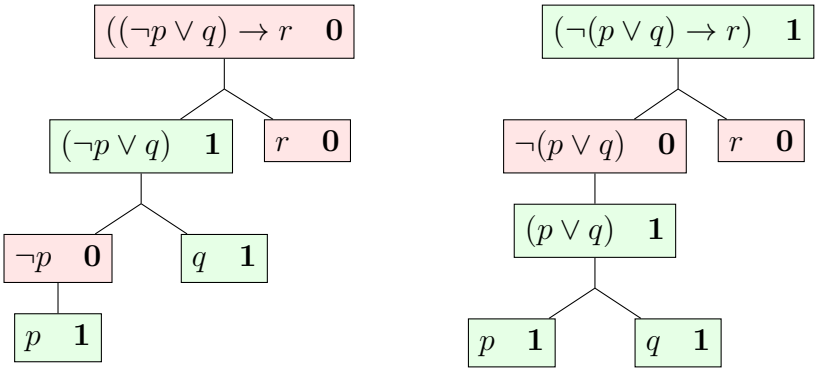
(2.18)

Bold-face numbers give the truth values for all relevant combinations of argument values: four in the case of connectives with two arguments, two in the case of the connective with one argument, the negation.

Explanation The tables for negation, conjunction, disjunction, and equivalence are quite intuitive, but the same does not hold for the table for implication. The table for implication has generated perennial debate, since it does not match the word 'implies' in natural language very well. E.g., does having a false *antecedent* (condition) φ and a true *consequent* ψ really make the implication if-φ-then-ψ true? But we are just doing the best we can in our simple two-valued setting. Here is a thought that has helped many students. You will certainly accept the following assertion as true: 'All numbers greater than 13 are greater than 12'. Put differently, 'if a number n is greater than 13 (p), then n is greater than 12 (q)'. But now, just fill in different numbers n, and you get all combinations in the truth table. For instance, $n = 14$ motivates the truth-value 1 for $p \to q$ at pq, $n = 13$ motivates 1 for $p \to q$ at $\overline{p}q$, and $n = 12$ motivates 1 for $p \to q$ at $\overline{pq}$.

A mismatch with natural language can actually be very useful. Conditionals are a 'hot spot' in logic, and it is a challenge to create systems that get closer to their behaviour. Propositional logic is the simplest treatment that exists, but other logical systems today deal with further aspects of conditionals in natural language and ordinary reasoning. You will see a few examples later in this course.

Computing truth tables for complex formulas How exactly can we compute truth values for complex formulas? This is done using our tables by following the construction stages of syntax trees. Here is how this works. Take the valuation V with $V(p) = V(q) = 1, V(r) = 0$ and consider two earlier formulas:



Incidentally, this difference in truth value explains our earlier point that these two variant formulas are different readings of the earlier natural language sentence.

Computing in this manner for all valuations, we can systematically tabulate the truth value

behaviour of complex propositional formulas in all relevant situations:

p	q	r	$((\neg p \vee q) \rightarrow r)$	$(\neg(p \vee q) \rightarrow r)$
0	0	0	**0**	**0**
0	0	1	**1**	**1**
0	1	0	**0**	**1**
0	1	1	**1**	**1**
1	0	0	**1**	**1**
1	0	1	**1**	**1**
1	1	0	**0**	**1**
1	1	1	**1**	**1**

(2.19)

Paying attention to the proper placement of brackets in formulas, you can compute truth-tables step by step for all situations. As an example we take the second formula from (2.19). First, start with summing up the situations and copy the truth-values under the proposition letters as has been done in the following table.

p	q	r	$(\neg$	$(p$	$\vee$	$q)$	$\rightarrow$	$r)$
0	0	0	$\cdot$	0	$\cdot$	0	$\cdot$	0
0	0	1	$\cdot$	0	$\cdot$	0	$\cdot$	1
0	1	0	$\cdot$	0	$\cdot$	1	$\cdot$	0
0	1	1	$\cdot$	0	$\cdot$	1	$\cdot$	1
1	0	0	$\cdot$	1	$\cdot$	0	$\cdot$	0
1	0	1	$\cdot$	1	$\cdot$	1	$\cdot$	1
1	1	0	$\cdot$	1	$\cdot$	0	$\cdot$	0
1	1	1	$\cdot$	1	$\cdot$	1	$\cdot$	1

(2.20)

Then start filling in the truth-values for the first possible operator. Here it is the disjunction: it can be computed because the values of its arguments are given (you can also see this from the construction tree). $(p \vee q)$ gets value 0 if and only if both p and q have the value 0. The intermediate result is given in the first table in (2.21). The next steps are the

negation and then the conjunction. This gives the following results:

$(\neg$	$(p$	$\vee$	$q)$	$\rightarrow$	$r)$
$\cdot$	0	0	0	$\cdot$	0
$\cdot$	0	0	0	$\cdot$	1
$\cdot$	0	1	1	$\cdot$	0
$\cdot$	0	1	1	$\cdot$	1
$\cdot$	1	1	0	$\cdot$	0
$\cdot$	1	1	0	$\cdot$	1
$\cdot$	1	1	1	$\cdot$	0
$\cdot$	1	1	1	$\cdot$	1

$(\neg$	$(p$	$\vee$	$q)$	$\rightarrow$	$r)$
1	0	0	0	$\cdot$	0
1	0	0	0	$\cdot$	1
0	0	1	1	$\cdot$	0
0	0	1	1	$\cdot$	1
0	1	1	0	$\cdot$	0
0	1	1	0	$\cdot$	1
0	1	1	1	$\cdot$	0
0	1	1	1	$\cdot$	1

$(\neg$	$(p$	$\vee$	$q)$	$\rightarrow$	$r)$
1	0	0	0	**0**	0
1	0	0	0	**1**	1
0	0	1	1	**1**	0
0	0	1	1	**1**	1
0	1	1	0	**1**	0
0	1	1	0	**1**	1
0	1	1	1	**1**	0
0	1	1	1	**1**	1

(2.21)

One does not have to draw three separate tables. All the work can be done in a single table. We just meant to indicate the right order of filling in truth-values.

Exercise 2.11 Construct truth tables for the following formulas:

- $(p \rightarrow q) \vee (q \rightarrow p)$,
- $((p \vee \neg q) \wedge r) \leftrightarrow (\neg(p \wedge r) \vee q)$.

Exercise 2.12 Using truth tables, investigate all formulas that can be readings of

$$\neg p \rightarrow q \vee r$$

(by inserting brackets in appropriate places), and show that they are not equivalent.

If, Only If, If and Only If Here is a useful list of different ways to express implications:

If p then q	$p \rightarrow q$
p if q	$q \rightarrow p$
p only if q	$p \rightarrow q$

The third item on this list may come as a surprise. To see that the third item is correct, reflect on how one can check whether "We will help you only if you help us" is false. This can can happen only if "We help you" is true, but "You help us" is false.

These uses of 'if' and 'only if' explain the use of the common abbreviation 'if and only if' for an equivalence. "We will help you if and only if you help us" states that "you help us" implies "we help you", and vice versa. A common abbreviation for 'if and only if' that we will use occasionally is *iff*.

2.6 Valid Consequence and Consistency

We now define the general notion of valid consequence for propositional logic. It is a more precise version of the notion of a valid argument that we introduced on page 2-4.

The notion runs over all possible valuations, and as we will see in a moment, we can use truth tables to check given inferences for validity. (In what follows, k can be any number. If it is $k = 0$, then there are no premises.)

Definition 2.13 (Valid consequence) The inference from a finite set of premises

$$\varphi_1, \ldots, \varphi_k$$

to a conclusion ψ is a *valid consequence*, something for which we write

$$\varphi_1, \ldots, \varphi_k \models \psi,$$

if each valuation V with $V(\varphi_1) = \ldots = V(\varphi_k) = 1$ also has $V(\psi) = 1$.

Definition 2.14 (Logical equivalence) If $\varphi \models \psi$ and $\psi \models \varphi$ we say that φ and ψ are *logically equivalent*.

Here it is useful to recall a warning that was already stated above. Do not confuse valid consequence with truth of formulas in a given situation: validity quantifies over truth in many situations, but it has no specific claim about truth or falsity of the premises and conclusions in the situation. Indeed, validity rules out surprisingly little in this respect: of all the possible truth/falsity combinations that might occur for premises and conclusion, it only rules out one case: viz. that all φ_i get value 1, while ψ gets value 0.

Another point from Section 2.2 that is worth repeating here concerns the role of propositional inference in conversation and argumentation. Valid inference does not just help establish truth, but it can also achieve a refutation of claims: when the conclusion of a valid consequence is false, at least one of the premises must be false. But logic does not tell us in general which one: some further investigation may be required to find the culprit(s). It has been said by philosophers that this refutational use of logic may be the most important one, since it is the basis of *learning*, where we constantly have to give up current beliefs when they contradict new facts.

Here is a simple example of how truth tables can check for validity:

Example 2.15 (Modus Tollens) The simplest case of refutation depends on the rule of *modus tollens*:

$$\varphi \to \psi, \neg\psi \models \neg\varphi.$$

Below you see the complete truth table demonstrating its validity:

φ	ψ	$\varphi \to \psi$	$\neg\psi$	$\neg\varphi$	
1	1	**1**	**0**	**0**	
1	0	**0**	**1**	**0**	
0	1	**1**	**0**	**1**	
0	0	**1**	**1**	**1**	!!

(2.22)

Of the four possible relevant situations here, only one satisfies both premises (the valuation on the fourth line), and we can check that there, indeed, the conclusion is true as well. Thus, the inference is valid.

By contrast, when an inference is invalid, there is at least one valuation (i.e., a line in the truth table) where its premises are all true, and the conclusion false. Such situations are called *counter-examples*. The preceding table also gives us a counter-example for the earlier invalid consequence

from $\varphi \to \psi, \neg\varphi$ to $\neg\psi$

namely, the valuation on the third line where $\varphi \to \psi$ and $\neg\varphi$ are true but $\neg\psi$ is false.

Please note that invalidity does not say that *all* valuations making the premises true make the conclusion false. The latter would express a valid consequence again, this time, the 'refutation' of ψ (since $\neg\varphi$ is true iff φ is false):

$$\varphi_1, \ldots, \varphi_k \models \neg\psi \tag{2.23}$$

Satisfiability Finally, here is another important logical notion that gives another perspective on the same issues:

Definition 2.16 (Satisfiable) A set of formulas X (say, $\varphi_1, \ldots, \varphi_k$) is *satisfiable* if there is a valuation that makes all formulas in X true.

There is a close connection between *satisfiability* and *consistency*.

Satisfiable versus Consistent A set of formulas that does not lead to a contradiction is called a *consistent* formula set. Here 'leading to a contradiction' refers to proof rules, so this is a definition in terms of proof theory. But it is really the other side of the same coin, for a set of formulas is consistent iff the set is satisfiable. Satisfiability gives the semantic perspective on consistency.

Instead of 'not consistent' we also say *inconsistent*, which says that there is no valuation where all formulas in the set are true simultaneously.

Satisfiability (consistency) is not the same as truth: it does not say that all formulas in X are actually true, but that they could be true in some situation. This suffices for many purposes. In conversation, we often cannot check directly if what people tell us is true (think of their accounts of their holiday adventures, or the brilliance of their kids), but we often believe them as long as what they say is consistent. Also, as we noted in Chapter 1, a lawyer does not have to prove that her client is innocent, she just has to show that it is consistent with the given evidence that he is innocent.

We can test for consistency in a truth table again, looking for a line making all relevant formulas true. This is like our earlier computations, and indeed, validity and consistency are related. For instance, it follows directly from our definitions that

$$\varphi \models \psi \text{ if and only if } \{\varphi, \neg\psi\} \text{ is not consistent.} \tag{2.24}$$

Tautologies Now we look briefly at the 'laws' of our system:

Definition 2.17 (Tautology) A formula ψ that gets the value 1 in every valuation is called a *tautology*. The notation for tautologies is $\models \psi$.

Many tautologies are well-known as general laws of propositional logic. They can be used to infer quick conclusions or simplify given assertions. Here are some useful tautologies:

$$\begin{array}{ll} \text{Double Negation} & \neg\neg\varphi \leftrightarrow \varphi \\ \text{De Morgan laws} & \neg(\varphi \vee \psi) \leftrightarrow (\neg\varphi \wedge \neg\psi) \\ & \neg(\varphi \wedge \psi) \leftrightarrow (\neg\varphi \vee \neg\psi) \\ \text{Distribution laws} & (\varphi \wedge (\psi \vee \chi)) \leftrightarrow ((\varphi \wedge \psi) \vee (\varphi \wedge \chi)) \\ & (\varphi \vee (\psi \wedge \chi)) \leftrightarrow ((\varphi \vee \psi) \wedge (\varphi \vee \chi)) \end{array} \tag{2.25}$$

Check for yourself that they all get values 1 on all lines of their truth tables.

Tautologies are a special zero-premise case of valid consequences, but via a little trick, they encode all valid consequences. In fact, every valid consequence corresponds to a tautology, for it is easy to see that:

$$\varphi_1, \ldots, \varphi_k \models \psi \text{ if and only if } (\varphi_1 \wedge \ldots \wedge \varphi_k) \rightarrow \psi \text{ is a tautology} \tag{2.26}$$

Exercise 2.18 Using a truth table, determine if the two formulas

$$\neg p \rightarrow (q \vee r), \neg q$$

together logically imply

(1) $p \wedge r$.

(2) $p \vee r$.

Display the complete truth table, and use it to justify your answers to (1) and (2).

Exercise 2.19

Show using a truth table that:

- the inference from $p \to (q \wedge r)$, $\neg q$ to $\neg p$ is valid and
- the inference from $p \to (q \vee r)$, $\neg q$ to $\neg p$ is not valid.

Exercise 2.20 Check if the following are valid consequences:

(1) $\neg(q \wedge r), q \models \neg r$

(2) $\neg p \vee \neg q \vee r, q \vee r, p \models r$.

Exercise 2.21 Give truth tables for the following formulas:

(1) $(p \vee q) \vee \neg(p \vee (q \wedge r))$

(2) $\neg((\neg p \vee \neg(q \wedge r)) \vee (p \wedge r))$

(3) $(p \to (q \to r)) \to ((p \to q) \to (p \to r))$

(4) $(p \leftrightarrow (q \to r)) \leftrightarrow ((p \leftrightarrow q) \to r)$

(5) $((p \leftrightarrow q) \wedge (\neg q \to r)) \leftrightarrow (\neg(p \leftrightarrow r) \to q)$

Exercise 2.22 Which of the following pairs are *logically equivalent*? Confirm your answer using truth tables:

(1) $\varphi \to \psi$ and $\psi \to \varphi$

(2) $\varphi \to \psi$ and $\neg\psi \to \neg\varphi$

(3) $\neg(\varphi \to \psi)$ and $\varphi \vee \neg\psi$

(4) $\neg(\varphi \to \psi)$ and $\varphi \wedge \neg\psi$

(5) $\neg(\varphi \leftrightarrow \psi)$ and $\neg\varphi \leftrightarrow \neg\psi$

(6) $\neg(\varphi \leftrightarrow \psi)$ and $\neg\varphi \leftrightarrow \psi$

(7) $(\varphi \wedge \psi) \leftrightarrow (\varphi \vee \psi)$ and $\varphi \leftrightarrow \psi$

2.7 Proof

Proof: symbolic inference So far we tested inferences for validity with truth tables, staying close to the semantic meaning of the formulas. But a lot of inference happens automatically, by manipulating symbols. People usually do not reason via truth tables. They rather combine many simple proof steps that they already know, without going back to their motivation. The more such rules they have learnt, the faster their reasoning goes. Likewise, mathematicians often do formal calculation and proof via symbolic rules (think of your school algebra), and of course, computers have to do proof steps purely symbolically (as long as they have not yet learnt to think, like us, about what their actions might mean).

Logic has many formal calculi that can do proofs, and later on, we will devote a whole chapter to this topic. But in this chapter, we give you a first taste of what it means to do proof steps in a formal calculus. There is a certain pleasure and surprise to symbolic calculation that has to be experienced.

Below, we present an *axiomatic system* organized a bit like the famous geometry book of Euclid's *Elements* from Antiquity. It starts from just a few basic principles (the axioms), after which chains of many proof steps, each one simple by itself, lead to more and more, sometimes very surprising theorems.

Here is a modern axiomatic symbol game for logic:

Definition 2.23 (Axiomatization) A *proof* is a finite sequence of formulas, where each formula is either an *axiom*, or follows from previous formulas in the proof by a deduction *rule*. A formula is a *theorem* if it occurs in a proof, typically as the last formula in the sequence. A set of axioms and rules defines an *axiomatization* for a given logic.
The following is an axiomatization for propositional logic. The axioms are given in schematic form, with the formula variables that we have already seen. It means that we can put any specific formula in the place of these variables:

(1) $(\varphi \to (\psi \to \varphi))$

(2) $((\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)))$

(3) $((\neg\varphi \to \neg\psi) \to (\psi \to \varphi))$

and there is only one deduction rule, the Modus Ponens that we have already encountered:

- if φ and $(\varphi \to \psi)$ are theorems, then ψ is also a theorem.

This axiomatization originates with the Polish logician Jan Łukasiewicz. In this system for propositional logic we may only use implication and negation symbols, and no other logical connectives, such as conjunctions. In our later section on expressivity it will be become clear why this restricted vocabulary is sufficient.

Training in axiomatic deduction will not be a key focus of this course. Still, we do want you to experience the interest of performing purely syntactic proofs, as a sort of 'symbol game' that can be interpreted later. We give one more abstract logical example here, and also one closer to practice.

Example 2.24 As an example of an axiomatic proof, we show that $p \to p$ is a theorem. This seems a self-evident tautology semantically, but now, the art is to derive it using only the rules of our game! In what follows we use well-chosen concrete instantiations of axiom schemas. For instance, the first line uses Axiom Schema 1 with the atomic proposition p for the variable formula φ and $q \to p$ for the variable formula ψ. And so on:

1. $p \to ((q \to p) \to p)$	Axiom (1)
2. $(p \to ((q \to p) \to p)) \to ((p \to (q \to p)) \to (p \to p))$	Axiom (2)
3. $(p \to (q \to p)) \to (p \to p)$	Modus Ponens, from steps $1, 2$
4. $p \to (q \to p)$	Axiom (1)
5. $p \to p$	Modus Ponens, from steps $3, 4$

It takes some skill to find such proofs by oneself. But it is actually an exciting game to many students, precisely because of the purely symbolic nature of the steps involved.

More general proofs can have certain assumptions, in addition to instances of axiom schemas. Here is an example closer to practice.

Example 2.25 Use only Modus Ponens and suitable axioms to derive the solution to the following problem. You want to throw a party, respecting people's incompatibilities. You know that:

(a) John comes if Mary or Ann comes.

(b) Ann comes if Mary does not come.

(c) If Ann comes, John does not.

Can you invite people under these constraints? There are several ways of solving this, including truth tables with update as in our next Section. But for now, can you prove what the solution must be? Here is a little help with the formal rendering:

> (i) 'If Ann comes, John does not' is the formula $a \to \neg j$, (ii) 'Ann comes if Mary does not come': $\neg m \to a$, (c) 'John comes if Mary or Ann comes': here you can rewrite to an equivalent conjunction 'John comes if Mary comes' and 'John comes if Ann comes' to produce two formulas that fall inside our language: $a \to j$, $m \to j$. Now try to give a proof just using the above axioms and rule for the solution, deriving successively that $\neg a$, m, j. Have fun!

This concludes our first glimpse of a proof game with a fixed repertoire.

System properties: soundness and completeness If all theorems of an axiomatic system are valid, the system is called *sound* , and conversely, if all valid formulas are provable theorems, the logic is called *complete*. Soundness seems an obvious requirement, as you want to rely totally on your proof procedure. The above system is sound, as you can see by noting that all axioms are tautologies, while Modus Ponens always takes tautologies to tautologies, that is, if φ and $\varphi \to \psi$ are tautologies, then ψ is also a tautology.

Completeness is a different matter, and can be harder to obtain for a given system. (Does Euclid's system of axioms suffice for proving all truths of geometry? The answer took centuries of investigation and reformulation of the system.) The above proof system is indeed complete, and so are the proof systems that we will present in later chapters. But showing that completeness holds can be hard. The completeness of predicate logic, that we will discuss in later chapters, was one of the first deep results in modern logic, discovered by the then 23-year old Kurt Gödel in his 1929 dissertation.

Axiomatic deduction is only one of many proof methods used in logic. Others include *natural deduction* (used a lot in logical Proof Theory) and *resolution* (used in many automated theorem provers). Chapter 9 in Part III of the book will take you much further into this area.

2.8 Information Update

With all this in place, we can now also define our earlier notions of *information* structure and information growth:

> The information content of a formula φ is the set $\mathsf{MOD}(\varphi)$ of its *models*, that is, the valuations that assign the formula φ the truth-value 1.

You can think of this as the range of possible situations that φ leaves open. Note that the more possibilities are left open by a formula φ, the less information φ contains. Formulas that leave many possibilities open correspond to information states with much uncertainty. Formulas that leave just one possibility open — that have just one satisfying valuation — leave no uncertainty at all about what the situation is like.

Information update by elimination of possibilities Here is the dynamics that changes such information states:

> An update with new information ψ reduces the current set of models X to the overlap or *intersection* of X and $\mathsf{MOD}(\psi)$. The valuations in X that assign the value 0 to ψ are *eliminated*.

Thus, propositional logic gives an account of basic cognitive dynamics, where information states (sets of satisfying valuations) shrink as new information comes in: growth of knowledge is loss of uncertainty.

We have seen earlier how this worked with simple inferences like

'from $p \vee q, \neg p$ to q',

if we assume that the premises update an initial information state of no information (maximal uncertainty: all valuations still present).

As a second example, we return to an earlier question in Section 2.3 (see Exercise 2.3)

What information is given by $p \vee q, \neg p \vee r$?

Here are the update stages:

$$\begin{array}{ll} \text{initial state} & \{pqr, pq\overline{r}, p\overline{q}r, p\overline{qr}, \overline{p}qr, \overline{p}q\overline{r}, \overline{pq}r, \overline{pqr}\} \\ \text{update with } p \vee q & \{pqr, pq\overline{r}, p\overline{q}r, p\overline{qr}, \overline{p}qr, \overline{p}q\overline{r}\} \\ \text{update with } \neg p \vee r & \{pqr, p\overline{q}r, \overline{p}qr, \overline{p}q\overline{r}\} \end{array} \tag{2.27}$$

We can conclude whatever is true in all of the remaining four states. One valid conclusion is the inclusive disjunction $q \vee r$, and this is indeed the one used in the so-called *resolution rule* of many automated reasoning systems. But actually, the two given premises are stronger than the inference $q \vee r$. The situation $pq\overline{r}$ is not among the ones that are left after the updates in (2.27), but $q \vee r$ is obviously true in this situation. One trivial way of really getting all content of the premises is of course just their conjunction:

$$(p \vee q) \wedge (\neg p \vee r). \tag{2.28}$$

But there is alo a disjunctive form that precisely describes the final information set $\{pqr, p\overline{q}r, \overline{p}qr, \overline{p}q\overline{r}\}$:

$$(p \wedge r) \vee (\neg p \wedge q). \tag{2.29}$$

In practice, we want convenient descriptions of information states, and later on we will look at some principles of Boolean Algebra that can help us with this.

Planning Other information scenarios arise in planning problems. Recall the Party Problem in Section 2.7 (Example 2.25). Can you invite people under the three constraints? One sure way is computing information updates from an initial state of no information about constraints:

$$\{maj, ma\overline{j}, m\overline{a}j, m\overline{a}\overline{j}, \overline{m}aj, \overline{m}a\overline{j}, \overline{ma}j, \overline{ma}\overline{j}\} \tag{2.30}$$

Now the three given premises update this initial information state, by removing options incompatible with them. In successive steps, (a), (b), (c) give the following reductions:

$$\begin{array}{lll} \text{(a)} & (m \vee a) \to j & \{maj, m\bar{a}j, \overline{m}aj, \overline{m}\overline{a}j, \overline{m}\overline{a}\overline{j}\} \\ \text{(b)} & \neg m \to a & \{maj, m\bar{a}j, \overline{m}aj\} \\ \text{(c)} & a \to \neg j & \{m\bar{a}j\} \end{array} \tag{2.31}$$

Incidentally, this is a unique solution for the stated constraints – but this need not at all be the case in general: there could be none, or more than one option remaining, too.

Games as information processing Our update process describes the information flow in games like *Master Mind*, where players have to guess the correct position of some hidden coloured pegs. In each round, she can make a guess, that gets evaluated by black marks for colours in correct positions, and white marks for colours that do occur, but placed in wrong positions.

For instance, let there be four possible colours red, white, blue, orange, and three positions, with a hidden correct sequence red-white-blue. Here is a table for a possible run of the game, indicating the information game in successive answers:

guess	answer	possibilities remaining
START		24
red, orange, white	•○	6
white, orange, blue	•○	2
blue, orange, red	○○	1

You will find it useful to do the updates, and see why the given numbers are correct.

Master Mind is not really interactive (a machine could provide the answers to your guesses), though new interactive variants are used these days in psychological experiments about children's reasoning with different agents. Information update with different agents, as well as more realistic games will be studied in Chapters 5 and 7. Elimination of possibilities is still fundamental there, so what you learnt here has a broad thrust.

2.9 Expressiveness

A logical language is not just an auxiliary tool for studying inferences and updates. It is also a language that can be used for the common things we have languages for: stating truths (and lies) about situations, communicating important facts to others, and so on. In this light, a very fundamental issue about a logical language is its *expressiveness*. What can we say with it?

For a start, propositional logic may look very poor. We can combine sentences, but we cannot look 'inside' them: "Horatio Nelson died at Trafalgar" is just an atomic proposition, say p. But the real issue how well it does within its own compass. And then we find a pleasant surprise:

Propositional logic is quite expressive! In total, there are sixteen possible Boolean operations (truth value assignments) with two arguments: count options in the truth table. This is many more than the number of binary operators we have in our language. Some of these correspond to serious expressions in natural language. In particular, the *exclusive disjunction* $\varphi \oplus \psi$ corresponds to the natural language phrasing 'either-φ-or-ψ'. It has the following truth table, compare it to the one for disjunction $\vee$:

φ	ψ	$\varphi \oplus \psi$	$\varphi \vee \psi$
0	0	**0**	**0**
0	1	**1**	**1**
1	0	**1**	**1**
1	1	**0**	**1**

Now note that we could get the same truth table by *defining* exclusive disjunction $\varphi \oplus \psi$ in terms of notions that we already had:

$$(\varphi \vee \psi) \wedge \neg(\varphi \wedge \psi) \text{ or, alternatively } \neg(\varphi \leftrightarrow \psi) \tag{2.32}$$

More generally, it is not hard to prove that

> All sixteen possible binary propositional operations are *definable*
> in terms of just the three operations $\neg, \wedge$ and $\vee$.

As an illustration,

> the implication $\varphi \rightarrow \psi$ has the same truth table as $\neg\varphi \vee \psi$ and as $\neg(\varphi \wedge \neg\psi)$.

In fact, even $\neg, \wedge$ alone suffice for defining all possible operations, and also $\neg, \vee$ alone, and $\neg, \rightarrow$ alone. As you will recall, the latter fact was used in the axiomatization of propositional logic in the section on proof.

Exercise 2.26 Define all connectives in terms of $\neg$ and $\wedge$.

Exercise 2.27 Define all connectives in terms of $\neg$ and $\rightarrow$.

Indeed, there is even an operation that can define all propositional operations by itself, the *Sheffer stroke*

$$\varphi \mid \psi,$$

defined as $\neg\varphi \vee \neg\psi$.

Now you know how expressive our language is on its own turf: it can express anything we want to say about combination of two-valued propositions.

This is just one of many interesting features of definability in propositional logic. Here is another, that we state without giving details. Every propositional logical formula, no matter how complex, is equivalent to a conjunction of disjunctions of proposition letters or their negations. This is called the 'conjunctive normal form' (there is also a disjunctive normal form). For instance, the conjunctive normal form for the earlier exclusive disjunction is

$$(\varphi \vee \psi) \wedge (\neg\varphi \vee \neg\psi). \tag{2.33}$$

Seeking a balance Clearly, there are many things that we cannot express in propositional logic. The following chapters are about more expressive languages, such as predicate logic or epistemic logic. Even so, an important thing to keep in mind is a *Balance*. In logic as in science, the art is to stay as simple as possible: 'Small is Beautiful'. A poor language may have special properties that make it elegant or useful. Propositional logic is very successful in bringing out basic reasoning patterns, and moreover, its very poverty leads to elegant and simple semantics and proof methods. In richer systems, the latter become more baroque, and sometimes essentially more complex.

This completes the standard part of this chapter. Next comes a sequence of special topics that will help you see where propositional logic lives in a larger scientific world.

2.10 Outlook — Logic, Mathematics, Computation

Studying logical phenomena via mathematical systems has proved a powerful method historically. Thinking about our language of logical forms yields general insights into expressive power, as we have just learnt. But also, thinking about a system of all validities per se yields new insights that can be used in many settings. Here are some examples:

Boolean algebra The system of laws shows many beautiful regularities. For instance, De Morgan and Distribution laws came in pairs with conjunction and disjunction interchanged. This 'duality' is general, and it reflects the close analogy between propositional

logic and binary arithmetic (arithmetic with just 0 and 1, where every number is represented in the binary, or base-2, number system). In particular, the truth tables are just laws of binary arithmetic when we read:

> $\vee$ as the maximum of two numbers, $\wedge$ as the minimum,
> and $\neg$ as flipping 0 and 1.

Suppressing details, distribution for conjunction over disjunction then matches the arithmetical distribution law $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$. But binary arithmetic is even better-behaved: it also validates another distribution law $x + (y \cdot z) = (x + y) \cdot (x + z)$ that does not hold for numbers in general (try some numbers, and you will see). We will pursue such connections between logic and computation in more detail in later chapters.

Abstraction and application Logical systems originally arose out of concrete practice. But conversely, once we have such abstract systems, new concrete interpretations may be found. Boolean algebra is an example. It describes a whole range of phenomena: propositional reasoning, binary arithmetic, reasoning with sets (where $\neg$ is complement, $\wedge$ intersection, and $\vee$ union), and even electrical switching circuits where conjunction is serial composition of networks, and disjunction is *parallel composition*. Thus, one and the same formula says lots of things!

For instance, consider one single abstract principle, the Boolean law of 'Absorption':

$$\varphi \leftrightarrow (\varphi \wedge (\varphi \vee \psi)) \tag{2.34}$$

This is a tautology for propositional reasoning that helps remove redundancies from discourse (or when used in the opposite direction, helping you make simple things sound complicated). Next, in binary arithmetic, it expresses a valid equation

$$x = x \min (x \max y) \tag{2.35}$$

about computing with minima and maxima. In set theory, Absorption is the valid principle

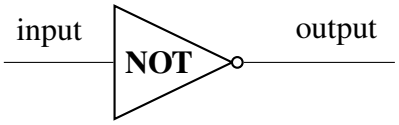
$$X = X \cap (X \cup Y) \tag{2.36}$$

which says that the intersection ('overlap') of the set X and the union of X and Y is the same as the set X itself.

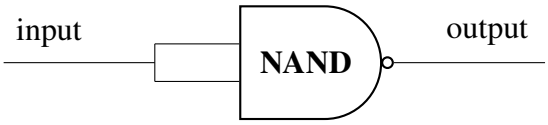
In a similar way, propositional logic plays a role in the design of logical (electronic) circuits. A NAND gate is an electronic circuit that behaves like the Sheffer stroke. The gate has two inputs and an output. If both inputs are 1 (carry a high voltage) then the output is low (carries a low voltage). For all other input combinations (high and low, low and high, low and low) the output is high. Since any propositional connective can be defined with just the Sheffer stroke, any desirable logical circuit can be built from a combination of NAND gates. Here is how negation is defined with the Sheffer stroke:

$$\varphi \mid \varphi.$$

The same principle can be used to build a NOT gate from a NAND gate:



NOT gate



NOT gate implemented with NAND gate

Thus we see a glimpse of a general reality: boolean algebra underlies real Boolean circuits in computers. The details of how this works can be found in many sources, including many sites on the internet.

Soundness and completeness The same general properties that we stated for our proof system also make sense here. In the nineteenth century, George Boole gave a complete algebraic analysis of propositional logic for reasoning with sentential operators like 'not', 'and', 'or', that has become famous as the 'Boolean algebra' that underlies the switching circuits of your computer. Here is what such a system looks like, with variables for propositions that can be true (1) or false (0), and operations $-$ for 'not', $\cdot$ for 'and', and $+$ for 'or' in the sense of binary arithmetic:

$$
\begin{array}{rclrcl}
x+(y+z) & = & (x+y)+z & x\cdot(y\cdot z) & = & (x\cdot y)\cdot z \\
x+y & = & y+x & x\cdot y & = & y\cdot x \\
x+x & = & x & x\cdot x & = & x \\
x+(y\cdot z) & = & (x+y)\cdot(x+z) & x\cdot(y+z) & = & (x\cdot y)+(x\cdot z) \\
x+(x\cdot y) & = & x & x\cdot(x+y) & = & x \\
-(x+y) & = & -x\cdot -y & -(x\cdot y) & = & -x+-y \\
x+0 & = & x & x\cdot 0 & = & 0 \\
x+1 & = & 1 & x\cdot 1 & = & x \\
x+-x & = & 1 & x\cdot -x & = & 0 \\
\multicolumn{6}{c}{--x \;=\; x}
\end{array}
\tag{2.37}
$$

It is easy to see that these equations correspond to valid tautologies, when read as equivalences between propositional formulas. Thus we have soundness: the calculus proves only valid principles. Conversely, Boolean algebra is also complete, and any valid equation can be derived from it by ordinary algebraic manipulations.

Computational complexity Propositional logic is tied up with computation in many ways, as we have seen in this chapter. In particular, truth tables make testing for logical validity a simple procedure that can be done mechanically. And indeed, there exist

computer programs for all of the tasks in this chapter. Within the compass of our simple language, this realizes a famous historical project: Leibniz's 'Calculus Ratiocinator' around 1700, which proposed that all reasoning can be reduced to computation. Indeed, there is a long history of 'logical machines' for carrying out inference tasks, going back to the Middle Ages.

Still, all this is computability in principle, and things are delicate in practice. A mechanical method with simple steps can still be highly complex when very many of these steps must be made. Consider truth tables. Computing truth values on a single line for a given formula goes fast. Earlier on we wrote successive truth values in the construction tree, and the number of time steps required for this is 'linear' (if the line has twice as many symbols, the computation takes roughly twice as many steps):

Computing a truth value for a formula takes linear time,

of the same order as the number of symbols in the formula. But now consider the whole truth table for a formula. With n atomic propositions we need 2^n lines, leading to *exponential growth* for the table in the size of the input:

Computing a truth table for validity takes exponential time.

This quickly outgrows the powers of even the fastest current computers. Therefore, smarter methods have been investigated, cutting down on the number of steps needed to test for validity — such as the semantic tableaus that you will see in Chapter 7. But it was always found that, in the worst case with difficult input formulas, these still require exponential time.

This is no coincidence. The exact computational complexity of validity in propositional logic is unknown: there may still be a faster method than existing ones that would work with a polynomial bound on processing time, though most experts doubt this. Determining this exact complexity is the essence of the famous

'$\mathbf{P} = \mathbf{NP}$ Problem' ,

that occurs on the famous 2000 Millennium List of open problems in mathematics posed by the Clay Mathematics Institute.

This problem is urgent since it has been shown that many basic computational tasks reduce to solving problems of validity and consistency in propositional logic. Thus, on its two-thousandth anniversary, propositional logic still poses deep problems.

Higher expressive power and undecidability Whether highly complex or not, the problem of testing for validity in propositional logic is *decidable*: there exists a mechanical method that computes the answer, at least in principle. Thus it may seem that computers can always do the job of logicians. But things change when we move to logics with higher expressive power, such as the predicate logic of Chapter 4 with quantifiers 'all', and 'some'. It is known from the work of Gödel, Turing, and others in the 1930s that there is no mechanical method at all for testing validity of predicate-logical inferences: these major systems pay a price for their greater expressive power: they are *undecidable*.

2.11 Outlook — Logic and Practice

The art of modelling To apply an abstract system like propositional logic, you need 'modelling skills'. For instance, we have already observed that you need to translate from natural language sentences to logical forms to get at the essence of an inference. This often takes practice, but it can be fun, witness the popular logic puzzles in commercial journals. Here is one simple example.

Propositional logic has generated many puzzles. The next exercise is from Raymond Smullyan's *The Lady or the Tiger?*, Penguin Books, 1982.

Exercise 2.28 Consider these two room signs:

- A – In this room there is a lady, and in the other one there is a tiger.
- B – In one of these rooms, there is a lady, and in one of them there is a tiger"

One of these signs is true, the other false. Behind which door is the lady?

But beyond this recreational aspect, propositional logic also applies to more serious areas of reasoning: witness, e.g., a whole literature on using propositional logic in legal reasoning. More technically, propositional logic has been applied to a wide variety of computational tasks, from Boolean circuits in your computer to complex train movements at the shunting yards of the Dutch Railways.

Such applications are not routine, and require creative skills.

Improving practice Training in propositional logic is also used to improve practical skills. This is an old tradition. Legend has it that medieval logic exams checked students' real-time skills as follows:

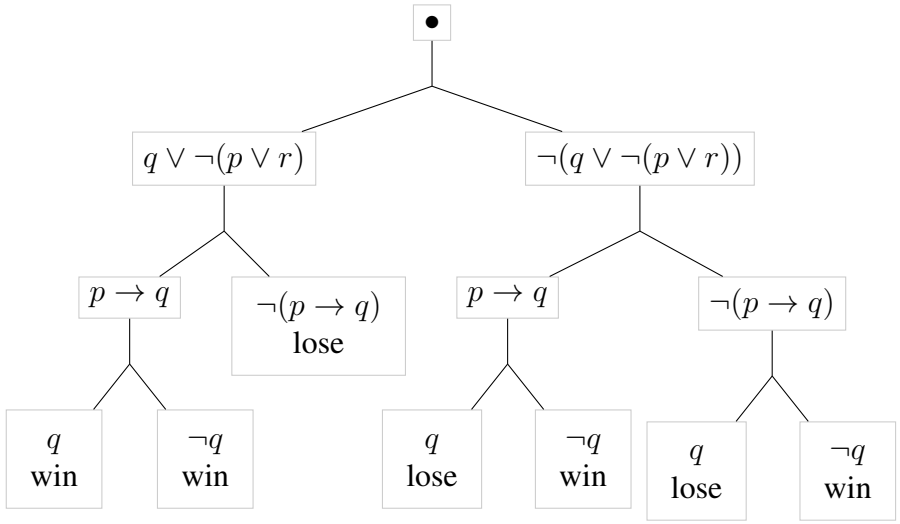
> *Obligatio Game* A finite number of rounds is chosen, the severity of the exam. The teacher gives the student successive assertions $\varphi_1, \ldots, \varphi_n$ that she has to 'accept' or 'reject' as they are put forward. In the former case, φ_i is added to

the students stock of commitments — in the latter, the negation $\neg\varphi_i$ is added.
The student passes if she maintains consistency throughout.

Suppose that a student is exposed to the following three statements:

$$(1)\ \ q \vee \neg(p \vee r), (2)\ \ p \to q, (3)\ \ q. \tag{2.38}$$

Here is one possible run. If you say YES to (1), you must say YES to (2), since it follows but then you can say either YES or NO to (3), since it is independent. Next, if you say NO to (1), you can say either YES or NO to (2), but then, in both cases, you must say NO to (3), as it follows from the negation of (1). The whole picture is:



This may be viewed as a *game tree* with all possible plays including the winning branches. (A complete tree would include Teacher's choices of the next assertion from some given set — possibly influenced by what Student has answered so far.) Either way, the tree will show that, as is only fair on exams, the student has a winning strategy for this game of consistency management. The logical reason is this:

Any consistent set of assertions can always be consistently expanded with at least one of the propositions φ, $\neg\varphi$.

The winning strategy based on this seems to require consistency checking at each stage, a hard computational problem. A simpler strategy for the student is this:

choose one model beforehand (say, a valuation making each atom true),
and evaluate each incoming assertion there, giving the obvious answers.

2.12 Outlook — Logic and Cognition

But how do logical systems relate to our daily practice where we reason and try to make sense without consciously thinking about how we do it? One interface with reality has occurred a number of times now:

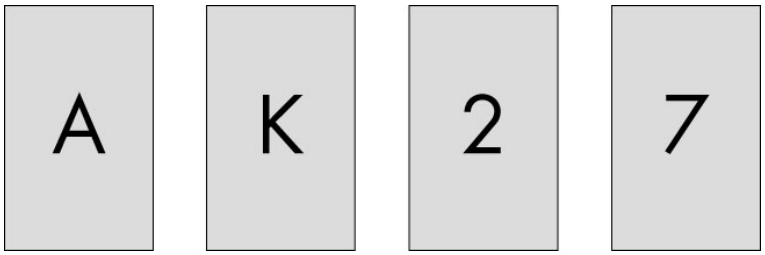
Logic and linguistics Natural languages have a much richer repertoire of meanings than the formal language of propositional logic. For instance, the expression *and* also has frequent non-Boolean readings. "John and Mary quarrelled" does not mean that "John quarrelled and Mary quarrelled", Likewise, we already noted that conditional expressions like *if ... then* do not behave exactly like the truth-table conditional. In particular, a false antecedent does not necessarily make them true: "If I were rich, I would be generous" does not follow from my not being rich.

But all this does not mean that logical methods do not apply. In fact, the divergence has turned a creative advantage. The richer structure of natural language has been an inexhaustible source of new logical theory. For instance, propositional logic has been generalized to work with more than two truth values to model vague or indeterminate uses of language, and the study of various sorts of conditional expressions has become a flourishing subdiscipline where logicians and linguists work together.

Logic and cognitive psychology The relation between logic and psychology has been somewhat stormier. It has been claimed by psychologists that everyday reasoning is highly non-logical. Here is a famous example.

The Wason selection task is a logic puzzle that states the following question:

> You are shown a set of four cards placed on a table each of which has a number on one side and a colored patch on the other side. The visible faces of the cards show 2, 7, A and K. Which card(s) should you turn over in order to test the truth of the proposition that if a card shows an even number on one face, then its opposite face shows a vowel?



The Wason selection task

Here is the correct response according to the logic of this chapter:

turn the cards showing 2 and K, but no other card.

The reason is this: to test the implication EVEN $\rightarrow$ VOWEL, we clearly need to check the card with the even number 2, but also should not forget the refutation case discussed several times before: if the card does not show a vowel, we need to make sure that it did not have an even number. Now the results of the experiment, repeated over many decades:

most people either (a) turn the 2 only, or (b) they turn the 2 and the A card.

Psychologists have suggested many explanations, including a 'confirmation bias' (refutation comes less natural to us) and an 'association bias' (red is mentioned so we check it). This seems to suggest that real reasoning is very different from what logic says.

However, the selection task tends to produce the correct logical response when presented in more concrete contexts that the experimental subjects are familiar with. For example, if the rule is 'If you are drinking alcohol, then you must be over 18', and the cards have an age on one side and a beverage on the other, e.g., '17', 'beer', '22', 'coke', most people have no difficulty in selecting the correct cards ('17' and 'beer'). Psychologists have used this as another argument against logic: the two settings have the same logical form, but very different behaviour results from familiarity effects.

More information on this famous experiment can be found on the webpage `http://en.wikipedia.org/wiki/Wason_selection_task`.

Frankly, all this polemics is not interesting. Clearly, people are not irrational, and if they ignored logic all the time, extracting the wrong information from the data at their disposal, it is hard to see how our species could survive. What seems to be the case is rather an issue of *representation* of reasoning tasks, and additional principles that play a role there. Moreover, the variation in outcomes fits with a conspicuous trend in modern logic, namely, the study of *task-dependent* forms of inference, whose rules may differ from the strict standards set in this chapter. These include more heuristic 'default rules' that are not valid in our strict sense, but that can be used until some problem arises that requires a revision of what we concluded so far.

But let us give the last word to George Boole, often considered the father of the purely mathematical approach to (propositional) logic. The title of his great work "The Laws of Thought" would seem to sit uneasily with a diehard normative mathematical perspective. But toward the end of the book, Boole remarks that he is serious about the title: the laws of propositional logic describe essential human thought. He also acknowledges that human reasoning often deviates from this canon. What that means is, he says, that there are *further laws* of human thought that still need to be discovered. That is what the modern interface of logic and cognitive science is about.

Further Exercises

Exercise 2.29 Prove that all propositional connectives are definable with the 'Sheffer stroke'

$$\varphi \mid \psi,$$

defined by $\neg\varphi \vee \neg\psi$.

Exercise 2.30 In how many ways can you win the following *obligatio* game?

$$(1)\ \ (p \to q) \vee (r \to q), (2)\ \ \neg((p \wedge r) \to q), (3)\ \ q.$$

Exercise 2.31 Consider the following formula:

$$(p \wedge (q \to r)) \to \neg(\neg p \vee ((\neg q \to q) \wedge (r \to \neg r))).$$

The logical symbols in this formula are all the symbols except parentheses and propositional variables. As you can see, the formula has 11 logical symbols. Answer the following questions:

(1) How many truth value entries does the truth table for this formula have. How does that number depend on the number of logical symbols?

(2) The truth table for a formula with 3 propositional variables has $2^3 = 8$ rows. How many entries in the truth table for such a formula do you have to compute (in the worst case) in order to find out if the formula is valid or not, given that you know that the formula has n logical symbols?

Summary *You have now seen your first logical system, and know how to reason in an exact mathematical manner with propositions. In particular, you have learnt these skills:*

- *read and write propositional formulas,*
- *translate simple natural language sentences into formulas,*
- *compute truth tables for various purposes,*
- *test validity of inferences,*
- *compute updates of information states,*
- *do some very simple formal proofs.*

In addition, you now have a working knowledge of the notions of

> *syntax, semantics, valuation, truth, valid consequence, tautology, consistency, axiomatic proof, expressive power, logical system.*

Finally, you have seen a first glimpse of connections between propositional logic and mathematical proof, computation, complexity, and some cognitive topics, namely, natural language and psychological experiments.

Further Reading Propositional logic was already known to the Stoic philosophers. See [Mat73] for an account. Propositional logic is fully developed in the famous book of George Boole [Boo54]. Boole gives an algebraic treatment of the logic of propositions. This kind of algebra is now known as Boolean algebra. A modern treatment of Boolean algebra is given in [GH09]. If you are in for some logic entertainment you should consult [CSI08] or the famous logic puzzle books by Raymond Smullyan [Smu09, Smu11].

Chapter 3

Syllogistic Reasoning

This chapter 'opens the box' of propositional logic, and looks further inside the statements that we make when we describe the world. Very often, these statements are about objects and their properties, and we will now show you a first logical system that deals with these. *Syllogistics* has been a standard of logical reasoning since Greek Antiquity. It deals with quantifiers like 'All P are Q' and 'Some P are Q', and it can express much of the common sense reasoning that we do about predicates and their corresponding sets of objects. You will learn a famous graphical method for dealing with this, the so-called 'Venn Diagrams', after the British mathematician John Venn (1834–1923), that can tell valid syllogisms from invalid ones. As usual, the chapter ends with some outlook issues, toward logical systems of inference, and again some phenomena in the real world of linguistics and cognition.

3.1 Reasoning About Predicates and Classes

Aristotle

John Venn

The Greek philosopher Aristotle (384 BC – 322 BC) proposed a system of reasoning in

his *Prior Analytics* (350 BC) that was so successful that it has remained a paradigm of logical reasoning for more than two thousand years: the *Syllogistic*.

Syllogisms A *syllogism* is a logical argument where a quantified statement of a specific form (the conclusion) is inferred from two other quantified statements (the premises).

The quantified statements are all of the form "Some/all A are B," or "Some/all A are not B," and each syllogism combines three predicates or properties. Notice that "All A are not B" can be expressed equivalently in natural language as "No A are B," and "Some A are not B" as "Not all A are B." We can see these quantified statements as describing relations between predicates, which is well-suited to describing hierarchies of properties. Indeed, Aristotle was also an early biologist, and his classifications of predicates apply very well to reasoning about species of animals or plants.

Your already know the following notion. A syllogism is called *valid* if the conclusion follows logically from the premises in the sense of Chapter 2: whatever we take the real predicates and objects to be: if the premises are true, the conclusion must be true. The syllogism is *invalid* otherwise.

Here is an example of a valid syllogism:

$$\frac{\begin{array}{l}\text{All Greeks are humans}\\ \text{All humans are mortal}\end{array}}{\text{All Greeks are mortal.}} \tag{3.1}$$

We can express the validity of this pattern using the $\models$ sign introduced in Chapter 2:

$$\text{All Greeks are humans, All humans are mortal} \models \text{All Greeks are mortal.} \tag{3.2}$$

This inference is valid, and, indeed, this validity has nothing to do with the particular predicates that are used. If the predicates *human*, *Greek* and *mortal* are replaced by different predicates, the result will still be a valid syllogism. In other words, it is the *form* that makes a valid syllogism valid, not the *content* of the predicates that it uses. Replacing the predicates by symbols makes this clear:

$$\frac{\begin{array}{l}\text{All A are B}\\ \text{All B are C}\end{array}}{\text{All A are C.}} \tag{3.3}$$

The classical quantifiers Syllogistic theory focusses on the quantifiers in the so called *Square of Opposition*, see Figure (3.1). The quantifiers in the square express relations between a first and a second predicate, forming the two arguments of the assertion. We

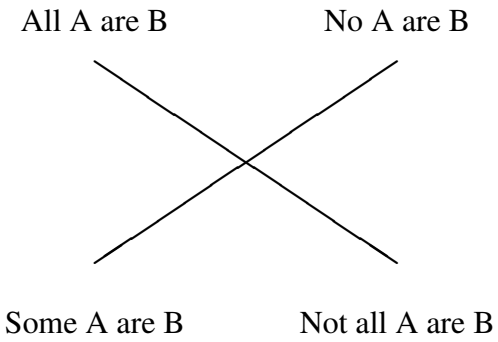


Figure 3.1: The Square of Opposition

think of these predicates very concretely, as sets of objects taken from some domain of discourse that satisfy the predicate. Say, 'boy' corresponds with the set of all boys in the relevant situation that we are talking about.

The quantified expressions in the square are related across the diagonals by external (sentential) negation, and across the horizontal edges by internal (or verb phrase) negation. It follows that the relation across the vertical edges of the square is that of internal plus external negation; this is the relation of so-called *quantifier duality*.

Because Aristotle assumes that the left-hand predicate A is non-empty (see below), the two quantified expressions on the top edge of the square cannot both be true; these expressions are called *contraries*. Similarly, the two quantified expressions on the bottom edge cannot both be false: they are so-called *subcontraries*.

Existential import Aristotle interprets his quantifiers with *existential import*: *All A are B* and *No A are B* are taken to imply that there are *A*. Under this assumption, the quantified expressions at the top edge of the square imply those immediately below them. The universal affirmative quantifier *all* implies the individual affirmative *some* and the universal negative *no* implies the individual negative *not all*. Existential import seems close to how we use natural language. We seldom discuss 'empty predicates' unless in the realm of phantasy. Still, modern logicians have dropped existential import for reasons of mathematical elegance, and so will we in this course.

The universal and individual affirmative quantifiers are said to be of types **A** and **I** respectively, from Latin **A***ff***I***rmo*, the universal and individual negative quantifiers of type **E** and **O**, from Latin *N***E***g***O**. Aristotle's theory was extended by logicians in the Middle Ages whose working language was Latin, whence this Latin mnemonics. Along these lines, *Barbara* is the name of the syllogism with two universal affirmative premises and a universal affirmative conclusion. This is the syllogism (3.1) above.

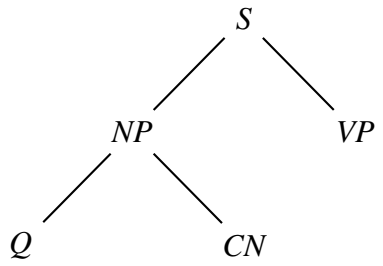
Here is an example of an invalid syllogism:

$$\frac{\begin{array}{l}\text{All warlords are rich}\\ \text{No students are warlords}\end{array}}{\text{No students are rich}} \tag{3.4}$$

Why is this invalid? Because one can picture a situation where the premises are true but the conclusion is false. Such a *counter-example* can be very simple: just think of a situation with just one student, who is rich, but who is not a warlord. Then the two premises are true (there being no warlords, all of them are rich – but you can also just add one rich warlord, if you like existential import). This 'picturing' can be made precise, and we will do so in a moment.

3.2 The Language of Syllogistics

Syllogistic statements consist of a quantifier, followed by a common noun followed by a verb: $Q\ N\ V$. This is an extremely general pattern found across human languages. Sentences S consist of a Noun Phrase *NP* and a Verb Phrase *VP*, and the Noun Phrase can be decomposed into a Determiner Q plus a Common Noun *CN*:



Thus we are really at the heart of how we speak. In these terms, a bit more technically, Aristotle studied the following inferential pattern:

$$\frac{\begin{array}{l}\text{Quantifier}_1\ \text{CN}_1\ \text{VP}_1\\ \text{Quantifier}_2\ \text{CN}_2\ \text{VP}_2\end{array}}{\text{Quantifier}_3\ \text{CN}_3\ \text{VP}_3}$$

where the quantifiers are *All*, *Some*, *No* and *Not all*. The common nouns and the verb phrases both express properties, at least in our perspective here ('man' stands for all men, 'walk' for all people who walk, etcetera). To express a property means to refer to a class of things, at least in a first logic course. There is more to predicates than sets of objects when you look more deeply, but this 'intensional' aspect will not occupy us here.

In a syllogistic form, there are two premises and a conclusion. Each statement refers to two classes. Since the conclusion refers to two classes, there is always one class that figures in the premises but not in the conclusion. The *CN* or *VP* that refers to this class is called the *middle term* that links the information in the two premises.

Exercise 3.1 What is the middle term in the syllogistic pattern given in (3.3)?

To put the system of syllogistics in a more systematic setting, we first make a brief excursion to the topic of operations on sets.

3.3 Sets and Operations on Sets

Building sets The binary relation $\in$ is called the element-of relation. If some object a is an element of a set A then we write $a \in A$ and if this is not the case we write $a \notin A$. Note that if $a \in A$, A is certainly a set, but a itself may also be a set. Example: $\{1\} \in \{\{1\}, \{2\}\}$.

If we want to collect all the objects together that have a certain property, then we write:

$$\{x \mid \varphi(x)\} \tag{3.5}$$

for the set of those x that have the property described by φ. Sometimes we restrict this property to a certain *domain of discourse* or *universe* U of individuals. To make this explicit, we write:

$$\{x \in U \mid \varphi(x)\} \tag{3.6}$$

to denote the set of all those x in U for which φ holds. Note that $\{x \in U \mid \varphi(x)\}$ defines a subset of U.

To describe a set of elements sharing multiple properties $\varphi_1, \ldots, \varphi_n$ we write:

$$\{x \mid \varphi_1(x), \ldots, \varphi_n(x)\} \tag{3.7}$$

Instead of a single variable, we may also have a sequence of variables. For example, we may want to describe a set of pairs of objects that stand in a certain relation. Here is an example.

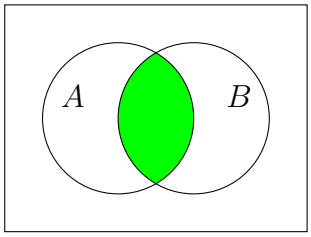
$$A = \{(x, y) \mid x \text{ is in the list of presidents of the US }, y \text{ is married to } x\} \tag{3.8}$$

For example, $(\text{Bill\_Clinton}, \text{Hillary\_Clinton}) \in A$ but, due to how the 2008 presidential election turned out, $(\text{Hillary\_Clinton}, \text{Bill\_Clinton}) \notin A$. Sets of pairs are in fact the standard mathematical representation of binary relations between objects (see Chapter A).

Operations on sets In talking about sets, one often also wants to discuss combinations of properties, and construct new sets from old sets. The most straightforward operation for this is the *intersection* of two sets:

$$A \cap B \quad = \quad \{x \mid x \in A \text{ and } x \in B\} \tag{3.9}$$

If A and B represent two properties then $A \cap B$ is the set of those objects that have both properties. In a picture:

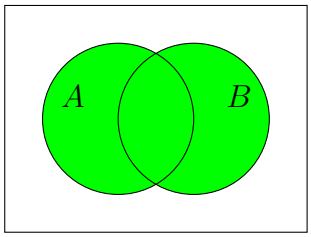


The intersection of the set of 'red things' and the set of 'cars' is the set of 'red cars'.

Another important operation is the *union* that represents the set of objects which have *at least one* of two given properties.

$$A \cup B \quad = \quad \{x \mid x \in A \text{ or } x \in B\} \tag{3.10}$$

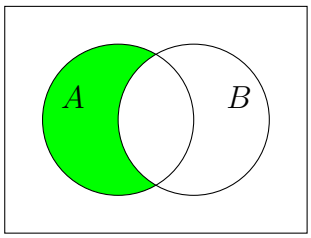
The 'or' in this definition should be read in the *inclusive* way. Objects which belong to both sets also belong to the union. Here is a picture:



A third operation which is often used is the *difference* of two sets:

$$A \setminus B \quad = \quad \{x \mid x \in A \text{ and } x \notin B\} \tag{3.11}$$

If we think of two properties represented by A and B then $A \setminus B$ represents those things that have the property A but not B. In a picture:

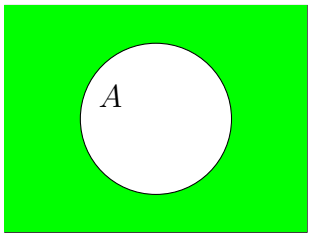


These pictorial representations of the set operations are called *Venn diagrams*, after the British mathematician John Venn (1834 - 1923). In a Venn diagram, sets are represented as circles placed in such a way that each combination of these sets is represented. In the case of two sets this is done by means of two partially overlapping circles. Venn diagrams are easy to understand, and interestingly, they are a method that also exploits our powers of non-linguistic visual reasoning.

Next, there is the *complement* of a set (relative to some given universe U (the domain of discourse):

$$\overline{A} \quad = \quad \{x \in U \mid x \notin A\} \tag{3.12}$$

In a picture:



Making use of complements we can describe things that do *not* have a certain property.

The complement operation makes it possible to define set theoretic operations in terms of each other. For example, the difference of two sets A and B is equal to the intersection of A and the complement of B:

$$A \setminus B \quad = \quad A \cap \overline{B} \tag{3.13}$$

Complements of complements give the original set back:

$$\overline{\overline{A}} \quad = \quad A \tag{3.14}$$

Complement also allows us to relate union to intersection, by means of the following so-called *de Morgan equations*:

$$\begin{aligned} \overline{A \cup B} &= \overline{A} \cap \overline{B} \\ \overline{A \cap B} &= \overline{A} \cup \overline{B} \end{aligned} \tag{3.15}$$

From the second de Morgan equation we can derive a definition of the union of two sets in terms of intersection and complement:

$$A \cup B = \overline{\overline{A}} \cup \overline{\overline{B}} = \overline{\overline{A} \cap \overline{B}} \tag{3.16}$$

This construction is illustrated with Venn diagrams in Figure 3.2. Also important are the so-called distributive equations for set operations; they describe how intersection distributes over union and vice versa:

$$\begin{aligned} A \cap (B \cup C) &= (A \cap B) \cup (A \cap C) \\ A \cup (B \cap C) &= (A \cup B) \cap (A \cup C) \end{aligned} \tag{3.17}$$

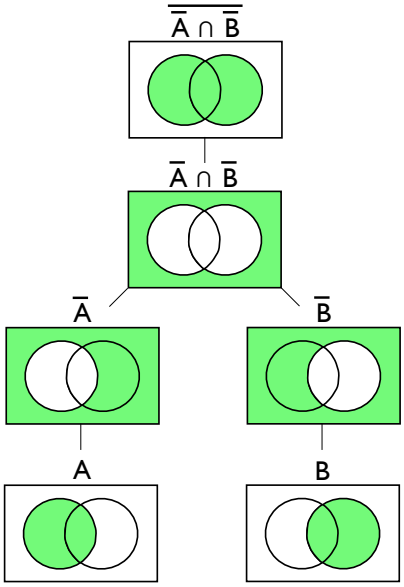


Figure 3.2: Construction of $A \cup B$ using intersection and complement.

Figure 3.3 demonstrates how the validity of the first of these equations can be computed by means of Venn-diagrams. Here we need three circles for the three sets A, B and C, positioned in such a graphical way that every possible combination of these three sets is represented in the diagrams.

The relation between sets and propositions The equalities between sets may look familiar to you. In fact, these principles have the same shape as propositional equivalences that describe the relations between $\neg$, $\wedge$ and $\vee$. In fact, the combinatorics of sets using complement, intersection and union *is* a Boolean algebra, where complement behaves like negation, intersection like conjunction and union like disjunction. The zero element of the algebra is the empty set $\emptyset$.

We can even say a bit more. The Venn-diagram constructions as in Figures 3.2 and 3.3 can be viewed as construction trees for set-theoretic expressions, and they can be reinterpreted as construction trees for formulas of propositional logic. Substitution of proposition letters for the base sets and replacing the set operations by the corresponding connectives gives a parsing tree with the corresponding semantics for each subformula made visible in the tree. A green region corresponds to a valuation which assigns the truth-value 1 to the given formula, and a white region to valuation which assigns this formula the value 0. You can see in the left tree given in Figure 3.3 that the valuations which makes the formula $a \wedge (b \vee c)$ true are abc, $a\bar{b}c$ and $ab\bar{c}$ (see Figure 3.4).

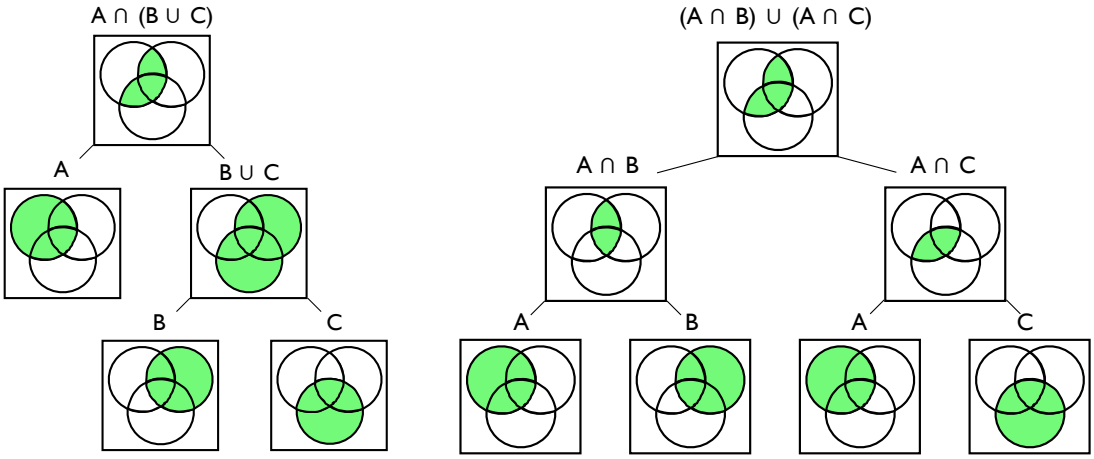


Figure 3.3: One of the distribution laws illustrated by means of Venn diagrams.

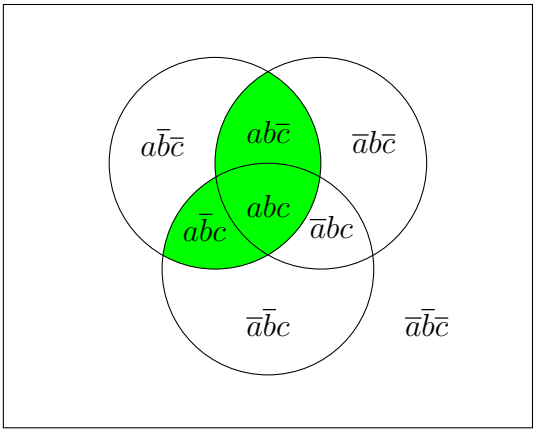
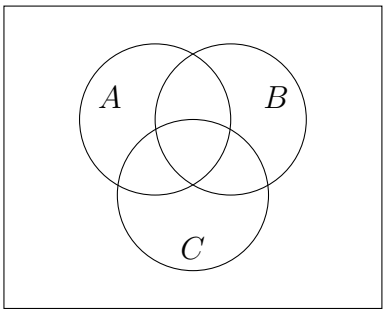


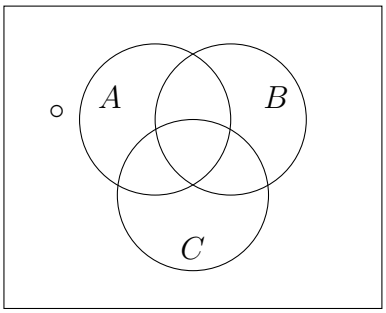
Figure 3.4: The support for $a \wedge (b \vee c)$ in a Venn-diagram.

3.4 Syllogistic Situations

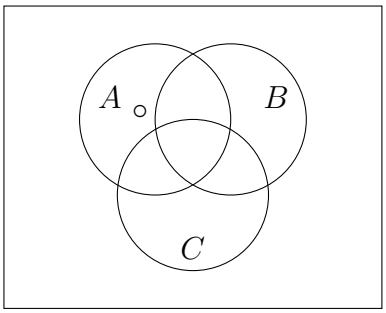
Since all syllogistic forms involve just three predicates A, B and C, we can draw a general picture of a syllogistic situation as the following *Venn Diagram*:



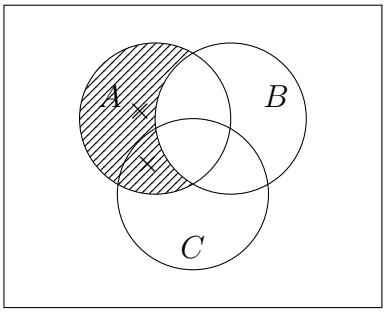
The rectangular box stands for a set of objects that form the domain of discourse, with three possible properties A, B and C. Note that there are 8 regions in all, quite properly, since that is the number of all possible combinations. An individual without any of these properties has to be outside of the three circles, like this:



An object with property A but lacking the properties B and C has to be inside the A circle, but outside the B and C circles, like this:

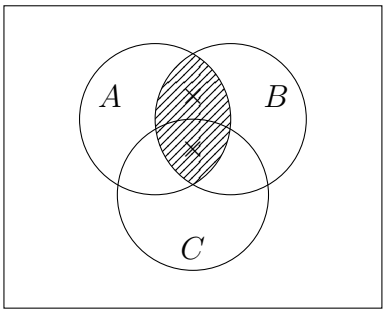


Now let us look in detail at what the Aristotelian quantifiers express. *All A are B* expresses that the part of the A circle outside the B circle has to be empty. We can indicate that in the picture by crossing out the forbidden regions, like this:



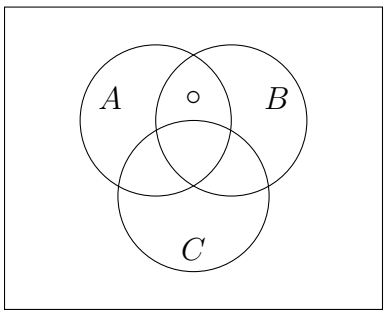
Note that the preceding picture does not take existential import into account. As we already said, we will leave it out in the interest of simplicity. And we lose nothing in this way. If you want to say that a predicate P is non-empty, you can always do so explicitly with a quantifier 'Some'.

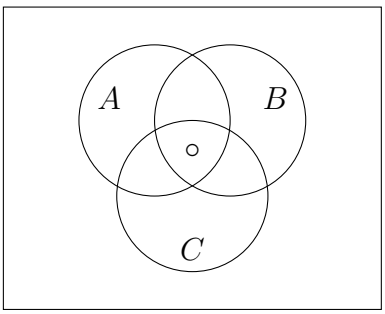
No A are B expresses that the part of the A circle that overlaps with the B circle has to be empty. Again, we can indicate this in a picture by crossing out the forbidden areas:



Again, existential import ("there must be A's") is not taken into account by this picture.

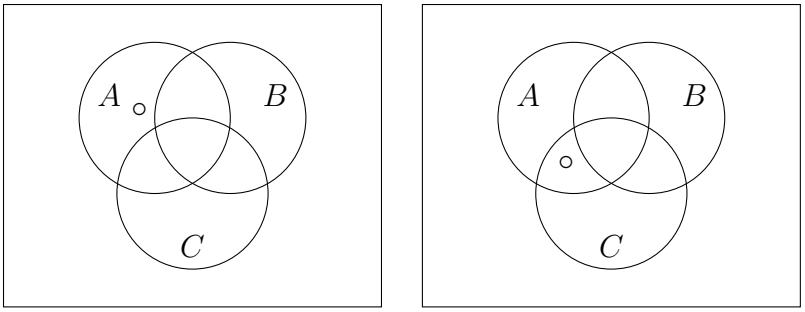
Now we move from universal quantifiers to existential ones. *Some A are B* expresses that the part of the picture where the A and the B circles overlap has to be non-empty. We can indicate that in the picture by putting an individual in an appropriate position. Since we do not know if that individual has property C or not, this can be done in two ways:





Not all A are B, or equivalently *Some are are not B*, expresses that the part of the A circle that falls outside the B circle has to be non-empty. There has to be at least one individual

that is an A but not a B. Since we do not know whether this individual has property C or not, we can again picture this information in two possible ways:



Some authors do not like this duplication of pictures, and prefer putting the small round circle for the individual on the border line of several areas.

You no doubt realize that such a duplication of cases makes the picture method much harder in terms of complexity, and hence, as we shall see, the art in checking validity for syllogisms is avoiding it whenever possible.

3.5 Validity Checking for Syllogistic Forms

The diagrams from the preceding section lead to a check for syllogistic validity:

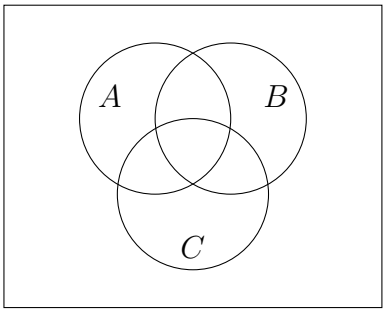
Working with diagrams We illustrate the method with the following valid syllogism:

$$\begin{array}{l} \text{All warlords are rich} \\ \text{No student is rich} \\ \hline \text{No warlord is a student} \end{array} \tag{3.18}$$

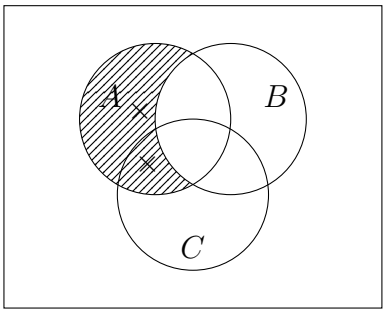
To carry out the validity check for this inference, we start out with the general picture of a domain of discourse with three properties. Next, we update the picture with the information provided by the premises. Here, the understanding is this:

> Crossing out a region with $\times$ means that this region is empty (there are no individuals in the domain of discourse with this combination of properties), while putting a $\circ$ in a region means that this region is non-empty (there is at least one individual with this combination of properties). Leaving a blank region means that there is no information about this region (there may be individuals with this combination of properties, or there may not).

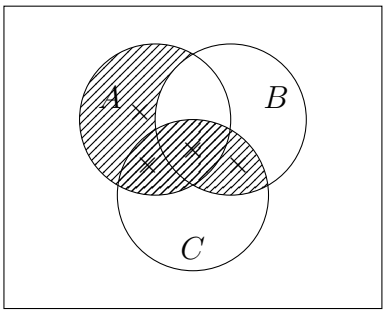
The method is this: we *update* with the information from the premises, and next *check* in the resulting picture whether the conclusion holds or not. Let A represent the property of being a warlord, B the property of being rich, and C the property of being a student. Then we start with the following general picture:



According to the first premise, *All A are B* has to be true, so we get:



By the second premise, *No C are B* has to be true, so we extend the picture as follows:



Finally, we have to check the conclusion. The conclusion says that the regions where A and C overlap have to be empty. Well, they are, for both of these regions have been crossed out. So the conclusion has to be true. Therefore, the inference is valid.

The general method The method we have used consists of the following four steps:

Draw the Skeleton Draw an empty picture of a domain of discourse with three properties A, B and C. Make sure that all eight combinations of the three sets are present.

Crossing out – Universal step Take the universal statements from the premises (the statements of the form "All ..." and "No ...", and cross out the forbidden regions in the diagram.

Filling up – Existential step Take the existential statements from the premises (the statements of the form "Some ..." and "Not all ..."), and try to make them true in the diagram by putting a $\circ$ in an appropriate region, while respecting the $\times$ signs. (This step might lead to several possibilities, all of which have to satisfy the check in the next item.)

Check Conclusion If the conclusion is universal it says that certain regions should have been crossed out. Are they? If the conclusion is existential it says that certain regions should have been marked with a $\circ$. Are they? If the answer to this question is affirmative the syllogism is valid; otherwise a counterexample can be constructed, indicating that the syllogism is invalid.

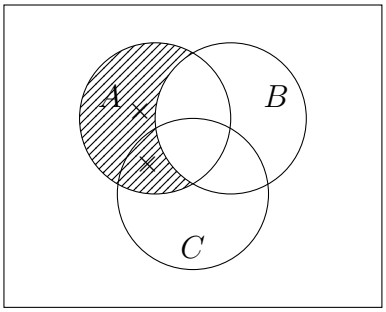
To illustrate the procedure once more, let us now take the invalid syllogism 3.4 that was mentioned before (repeated as 3.19).

$$\begin{array}{l} \text{All warlords are rich} \\ \text{No student is a warlord} \\ \hline \text{No student is rich} \end{array} \tag{3.19}$$

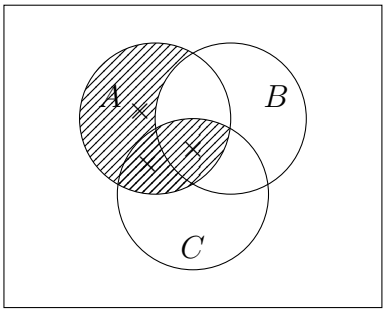
The symbolic form of this syllogism is:

$$\text{All A are B} \quad \text{No C are A} \quad \text{Therefore: No C are B.} \tag{3.20}$$

The premise statements are both universal. Crossing out the appropriate regions for the first premise gives us:



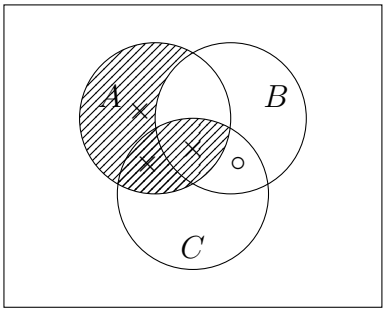
After also crossing out the regions forbidden by the second premise we get:



Note that the region for the AC's outside B gets ruled out twice. It looks like the second premise repeats some of the information that was already conveyed by the first premise (unlike the case with the previous example). But though this may say something about presentation of information, it does not affect valid or invalid consequences.

Finally, we check whether the conclusion holds. *No C are B* means that the regions where C and B overlap are forbidden. Checking this in the diagram we see that the region where A, B and C overlap is indeed crossed out, but the region outside A where B and C overlap is not. Indeed, the diagram does not contain information about this region. This means that we can use the diagram to construct a counterexample to the inference.

> The diagram allows us to posit the existence of an object that satisfies B and C but not A, in the concrete case of our example, a rich student who is not a warlord:



This final diagram gives the shape that all counterexamples to the valididity of 3.19 have in common. All these counterexamples will have no objects in the forbidden regions, and at least one object in the region marked with $\circ$.

Venn diagrams actually have a long history in logic, going back to the 18th century, and they are still an object of study in cognitive science, since they somehow combine visual and symbolic reasoning – a basic human ability that is not yet fully understood..

Exercise 3.2 Check the following syllogistism for validity, using the method just explained.

$$\frac{\begin{array}{l}\text{Some philosophers are Greek}\\ \text{No Greeks are barbarians}\end{array}}{\text{No philosophers are barbarians.}} \tag{3.21}$$

Exercise 3.3 Check the following syllogistic pattern for validity.

$$\frac{\begin{array}{l}\text{No Greeks are barbarians}\\ \text{No barbarians are philosophers}\end{array}}{\text{No Greeks are philosophers.}} \tag{3.22}$$

Exercise 3.4 Check the following syllogistic pattern for validity.

$$\frac{\begin{array}{l}\text{No Greeks are barbarians}\\ \text{Some barbarians are philosophers}\end{array}}{\text{Not all philosophers are Greek.}} \tag{3.23}$$

Exercise 3.5 Can you modify the method so that it checks for syllogistic validity, but now with the quantifiers all read with existential import? How?

More than three predicates What follows is a digression for the interested reader. Venn diagrams were a high point of traditional logic, just before modern logic started. How far does this method take us?

The validity check for syllogistics can be extended to inferences with more than two premises (and more than three predicates). This can still be done graphically (Venn had several beautiful visualizations), but you may also want to think a bit more prosaically in terms of tabulating possibilities. Here is one way (disregarding matters of computational efficiency).

For purposes of exposition, assume that four predicates A, B, C, D occur in the inference. List all possible combinations in a table (compare the tables for the propositonal variables in Chapter 2 – we economized a bit here, writing the property only when it holds):

	A	B	AB
C	AC	BC	ABC
D	AD	BD	ABD
CD	ACD	BCD	ABCD

Take as example the following entailment

$$\text{All A are B}, \text{No C are B}, \text{Some C are D}, \text{Therefore: Not all D are A}. \tag{3.24}$$

Again we can use the update method to check whether this is valid. First update with the information that all A are B. This rules out certain possibilities:

	A $\times$	B	AB
C	AC $\times$	BC	ABC
D	AD $\times$	BD	ABD
CD	ACD $\times$	BCD	ABCD

All A are B

The information that no C are B also rules out possibilities, as follows:

	A	B	AB
C	AC	BC $\times$	ABC $\times$
D	AD	BD	ABD
CD	ACD	BCD $\times$	ABCD $\times$

No C are B

Combining these two updates, we get:

	A $\times$	B	AB
C	AC $\times$	BC $\times$	ABC $\times$
D	AD $\times$	BD	ABD
CD	ACD $\times$	BCD $\times$	ABCD $\times$

All A are B and No C are B

The third premise, "some C are D," is existential. It states that there has to at least one CD combination in the table. There is only one possibility for this:

	A $\times$	B	AB
C	AC $\times$	BC $\times$	ABC $\times$
D	AD $\times$	BD	ABD
CD $\circ$	ACD $\times$	BCD $\times$	ABCD $\times$

Finally, we must check whether "not all D are A" holds in the table that results from updating with the premises. And indeed it does: region CD is non-empty (indicated by the presence of the $\circ$), so it gives us a witness of a D which is not an A. Therefore, the given inference must be valid.

The syllogistic system as such Working through the exercises of this section you may have realized that the diagrammatic validity testing method can be applied to any syllogism, and that, in the terms of Chapter 2:

> The syllogistic is *sound* (only valid syllogism pass the test)
> and *complete* (all valid syllogisms pass the test).

Moreover, the method decides the question of validity in a matter of a few steps. Thus, again in our earlier terms:

> The syllogistic is a *decision method* for validity,

the system of the Syllogistic is 'decidable'. This is like what we saw for propositional logic, and indeed, it can be shown that the two systems are closely related, though we shall not do so here.

Much more can be said about the history of the syllogistic. The website of this course has an improved version of the Venn Diagram method due to Christie Ladd in 1882 which shows how it can be turned into a more efficient 'refutation method' when we picture the premises, but also the *negation* of the conclusion, and then try to spot a contradiction.

As usual, the rest of this chapter explores a few connections with other areas, starting with mathematical systems, then moving to computation, and ending with cognition. These topics are not compulsory in terms of understanding all their ins and outs, but they should help broaden your horizon.

3.6 Outlook — Satisfiability and Complexity

The tabling method for testing the validity of syllogisms suggests that the method behaves like the truth table method for propositional logic: if there are n properties, the method checks 2^n cases. For propositional logic it is an open question whether a non-exponential method exists for checking satisfiability: this is the famous *P* versus *NP* problem. But how about syllogistics? Can we do better than exponential?

Focussing on universal syllogistic forms only, it is easy to see that a set of universal forms is always satisfiable, provided we forget about existential import. The reason for this is that a situation with all classes empty will satisfy any universal form. Therefore:

> A set of syllogistic forms consisting of only universal statements is always satisfiable.

And, as a straightword consequence of this:

> A syllogism with only universal premises and an existential conclusion is always invalid.

The reason for this is that the situation with all classes empty is a counterexample: it will satisfy all the premisses but will falsify the existential conclusion.

If you reflect on this you see that the unsatisfiability of a set of syllogistic forms Σ is always due to the absence of witnesses for some existential forms $\psi_1, ..., \psi_n$ in Σ. Now, since the *number* of witnesses for a particular property does not matter – one witness for some property is as good as many – we can limit attention to situations where there is just a single object in the universe:

> A finite set of syllogistic forms Σ is unsatisfiable if and only if there exists an existential form ψ such that ψ taken together with the universal forms from Σ is unsatifiable.

The interesting thing is that this restricted form of satisfiability can easily be tested with propositional logic, as follows. Remember that we are talking about the properties of a single object x. Let proposition letter a express that object x has property A. Then a universal statement "all A are B" gets translated into $a \rightarrow b$: if x has property A then x also has property B. An existential statement "some A are B" gets translated into $a \wedge b$, expressing that x has both properties A and B. The universal negative statement "no A are B" gets translated into $a \rightarrow \neg b$, and the negative existential statement "some A are not B" gets translated as $a \wedge \neg b$. The nice thing about this translation is that it employs a single proposition letter for each property. No exponential blow-up here.

Note that to test the satisfiability of a set of syllogistic statements containing n existential statements we will need n tests: we have to check for each existential statement whether it is satisfiable when taken together with all universal statements. But this does not cause exponential blow-up if all these tests can be performed efficiently. We will show now that they can.

It may look like nothing is gained by our translation to propositional logic, since all known general methods for testing satisfiability of propositional logical formulas are exponential. But the remarkable thing is that our translation uses a very well-behaved fragment of propositional logic, for which satisfiability testing is easy.

In this outlook, we briefly digress to explain how propositional logic can be written in clausal form, and how satisfiability of clausal forms can be tested efficiently, provided the forms are in a 'nice' shape. Here are some definitions:

literals a literal is a proposition letter or its negation. If l is a literal, we use $\bar{l}$ for its negation: if l has the form p, then $\bar{l}$ equals $\neg p$, if l has the form $\neg p$, then $\bar{l}$ equals p. So if l is a literal, then $\bar{l}$ is also a literal, with opposite sign.

clause a clause is a set of literals.

clause sets a clause set is a set of clauses.

Read a clause as a *disjunction* of its literals, and a clause set as a *conjunction* of its clauses.

Here is an example: the clause form of

$$(p \to q) \wedge (q \to r)$$

is

$$\{\{\neg p, q\}, \{\neg q, r\}\}.$$

And here is an inference rule for clause sets called **Unit Propagation**:

Unit Propagation If one member of a clause set is a singleton $\{l\}$ (a 'unit'), then:

(1) remove every other clause containing l from the clause set (for since l has to be true, we know these other clauses have to be true as well, and no information gets lost by deleting them);

(2) remove $\bar{l}$ from every clause in which it occurs (for since l has to be true, we know that $\bar{l}$ has to be false, so no information gets lost by deleting $\bar{l}$ from any disjunction in which it occurs).

The result of applying this rule is an equivalent clause set. Example: applying unit propagation using unit $\{p\}$ to

$$\{\{p\}, \{\neg p, q\}, \{\neg q, r\}, \{p, s\}\}.$$

yields:

$$\{\{p\}, \{q\}, \{\neg q, r\}\}.$$

Applying unit propagation to this, using unit $\{q\}$ yields

$$\{\{p\}, \{q\}, \{r\}\}.$$

The *Horn fragment* of propositional logic consists of all clause sets where every clause has at most one positive literal. HORNSAT is the problem of checking Horn clause sets for satisfiability. This check can be performed in polynomial time (linear in the size of the formula, in fact).

If unit propagation yields a clause set in which units $\{l\}, \{\bar{l}\}$ occur, the original clause set is unsatisfiable, otherwise the units in the result determine a satisfying valuation. Recipe: for any units $\{l\}$ occurring in the final clause set, map their proposition letter to the truth value that makes l true; map all other proposition letters to false.

> The problem of testing satisfiability of syllogistic forms containing exactly one existential statement can be translated to the Horn fragment of propositional logic.

To see that this is true, check the translations we gave above:

All A are B $\mapsto$ $a \rightarrow b$ or equivalently $\{\{\neg a, b\}\}$.

No A are B $\mapsto$ $a \rightarrow \neg b$ or equivalently $\{\{\neg a, \neg b\}\}$.

Some A are B $\mapsto$ $a \wedge b$ or equivalently $\{\{a\}, \{b\}\}$.

Not all A are B $\mapsto$ $a \wedge \neg b$ or equivalently $\{\{a\}, \{\neg b\}\}$.

As you can see, these translations are all in the Horn fragment of propositional logic. We conclude that satisfiability of sets of syllogistic forms can be checked in time polynomial in the number of properties mentioned in the forms.

Exercise 3.6 ♠ Consider the following three syllogisms:

No A are B	No A are B	All B are A
Not all B are C	Some B are C	Some C are A
Some C are A	Not all C are A	Some A are B

(1) One of the three syllogisms is valid. Which one?

(2) Use the diagram method to show the validity of the syllogism you claim is valid.

(3) Use a diagram to show that the other syllogisms are invalid.

(4) Next, show, for these three cases, how the validity of the syllogisms can be checked by translating the premisses and the negation of the conclusion into clausal form, and then using unit propagation to check the resulting clause set for satisfiability. (Note: the clause set is satisfiable *iff* the syllogism is invalid.)

3.7 Outlook — The Syllogistic and Actual Reasoning

Aristotle's system is closely linked to the grammatical structure of natural language, as we have said at the start. Indeed, many people have claimed that it stays so close to our ordinary language that it is part of the *natural logic* that we normally use. Medieval logicians tried to extend this, and found further patterns of reasoning with quantifiers that share these same features of staying close to linguistic syntax, and allowing for very simple inference rules. 'Natural Logic' is a growing topic these days, where one tries to find large simple inferential subsystems of natural language that can be described without too much mathematical system complexity. Even so, we have to say that the real logical hitting power will only come in our next chapter on predicate logic, which consciously

deviates from natural language to describe more complex quantifier reasoning of types that Aristotle did not handle.

Syllogistic reasoning has also drawn the attention of cognitive scientists, who try to draw conclusions about what goes on in the human brain when we combine predicates and reason about objects. As with propositional reasoning, one then finds differences in performance that do not always match what our methods say, calling attention to the issue how the brain represents objects and their properties and relations. From another point of view, the *diagrammatic* aspect of our methods has attracted attention from cognitive scientists lately. It is known that the brain routinely combines symbolic language-oriented and visual and diagrammatic representations, and the Venn Diagram method is one of the simplest pilot settings for studying how this combination works.

Summary *In this chapter you have learnt how one simple but very widespread kind of reasoning with predicates and quantifiers works. This places you squarely in a long logical tradition, before we move to the radical revolutions of the 19th century in our next chapter. More concretely, you are now able to*

- *write basic syllogistic forms for quantifiers,*
- *understand set diagram notation for syllogistic forms,*
- *test syllogistic inferences using Venn diagrams,*
- *understand how diagrams allow for update,*
- *understand connections with propositional logic,*
- *understand connections with data representation.*

Further Reading If you wish to be instructed in logic by the teacher of Alexander himself, you should consult the *Prior Analytics* [Ari89] (available online, in a different translation, at `classics.mit.edu/Aristotle/prior.html`). For a full textbook on Aristotelean logic, see [PH91].

Aristotelean logic can be viewed as a logic of concept description. See the first and second chapter [NB02, BN02] of the Description Logic Handbook [BCM$^+$02] for more information about this connection. Connections between Aristotelian logic and predicate logic (see next Chapter of this book) are discussed in [Łuk51]. Extensions of Aristotelian logic in the spirit of syllogistics are given in [PH04] and [Mos08].

Chapter 4

The World According to Predicate Logic

Overview At this stage of our course, you already know propositional logic, the system for reasoning with sentence combination, which forms the basic top-level structure of argumentation. Then we zoomed in further on actual natural language forms, and saw how sentences make quantified statements about properties of objects, providing a classification of the world in terms of a hierarchy of smaller or larger predicates. You also learnt the basics of syllogistic reasoning with such hierarchies.

In this Chapter, we look still more deeply into what we can actually say about the world. You are going to learn the full system of 'predicate logic' of objects, their properties, but also the relations between them, and about these, arbitrary forms of quantification. This is the most important system in logic today, because it is a *universal language* for talking about *structure*. A structure is any situation with objects, properties and relations, and it can be anything from daily life to science: your family tree, the information about you and your friends on Facebook, the design of the town you live in, but also the structure of the number systems that are used in mathematics, geometrical spaces, or the universe of sets. In the examples for this chapter, we will remind you constantly of this broad range from science to daily life.

Predicate logic has been used to increase precision in describing and studying structures from linguistics and philosophy to mathematics and computer science. Being able to use it is a basic skill in many different research communities, and you can find its notation in many scientific publications. In fact, it has even served as a model for designing new computer languages, as you will see in one of our Outlooks. In this chapter, you will learn how predicate logic works, first informally with many examples, later with more formal definitions, and eventually, with outlooks showing you how this system sits at the interface of many disciplines. But this power comes at a price. This chapter is not easy, and mastering predicate logic until it comes naturally to you takes a while – as successive generations of students (including your teachers) have found.

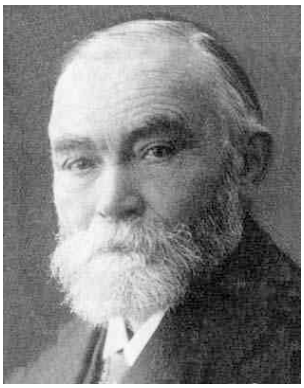
4.1 Learning the Language by Doing

Zooming in on the world Propositional logic classifies situations in terms of 'not', 'and', 'or' combinations of basic propositions. This truth-table perspective is powerful in its own way (it is the basis of all the digital circuits running your computer as you are reading this), but poor in other respects. Basic propositions in propositional logic are not assumed to have internal structure. "John walks" is translated as p, "John talks" as q, and the information that both statements are about John gets lost. Predicate logic looks at the internal structure of such basic facts. It translates "John walks" as Wj and "John talks" as Tj, making it clear that the two facts express two properties of the same person, named by the constant j.

As we said, predicate logic can talk about the internal structure of situations, especially, the objects that occur, properties of these objects, but also their relations to each other. In addition, predicate logic has a powerful analysis of universal quantification (all, every, each, ...) and existential quantification (some, a, ...). This brings it much closer to two languages that you already knew before this course: the natural languages in the common sense world of our daily activities, and the symbolic languages of mathematics and the sciences. Predicate logic is a bit of both, though in decisive points, it differs from natural language and follows a more mathematical system. That is precisely why you are learning something new in this chapter: an additional style of thinking.

Two founding fathers Predicate logic is a streamlined version of a "language of thought" that was proposed in 1878 by the German philosopher and mathematician Gottlob Frege (1848 – 1925). The experience of a century of work with this language is that, in principle, it can write all of mathematics as we know it today. Around the same time, essentially the same language was discovered by the American philosopher and logician Charles Saunders Peirce. Peirce's interest was general reasoning in science and daily life, and his ideas are still inspirational to modern areas philosophers, semioticists, and researchers in Artificial Intelligence. Together, these two pioneers stand for the full range of predicate logic.

Charles Sanders Peirce



Gottlob Frege

We will now introduce predicate logic via a sequence of examples. Grammar comes later: further on in this chapter we give precise grammatical definitions, plus other information.

> *If you are more technically wired, you can skim the next four introductory sections, and then go straight to the formal part of this chapter.*

We do not start in a vacuum here: the natural language that you know already is a running source of examples and, in some cases, contrasts:

The basic vocabulary We first need names for *objects*. We use constants ('proper names') a, b, c, ... for special objects, and variables x, y, z, ... when the object is indefinite. Later on, we will also talk about function symbols for complex objects.

Then, we need to talk about *properties and predicates* of objects. Capital letters are predicate letters, with different numbers of 'arguments' (i.e., the objects they relate) indicated. In natural language, 1-place predicates are intransitive verbs ("walk") and common nouns ("boy"), 2-place predicates are transitive verbs ("see"), and 3-place predicates are so-called ditransitive verbs ("give"). 1-place predicates are also called *unary predicates*, 2-place predicates are called *binary predicates*, and 3-place predicates are called *ternary predicates*. In natural language ternary predicates are enough to express the most complex verb pattern you can get, but logical languages can handle any number of arguments.

Next, there is still *sentence combination*. Predicate logic gratefully incorporates the usual operations from propositional logic: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$. But in addition, and very importantly, it has a powerful way of expressing *quantification*. Predicate logic has quantifiers $\forall x$ ("for all x") and $\exists x$ ("there exists an x") tagged by *variables* for objects, that can express an amazing number of things, as you will soon see.

From natural language to predicate logic For now, here is a long list of examples showing you the underlying 'logical form' of the statements that you would normally make when speaking or writing. Along the way we will point out various important features.

Atomic statements We start with the simplest statements about objects:

natural language	*logical formula*
John walks	Wj
John is a boy	Bj
He walks	Wx
John sees Mary	Sjm
John gives Mary the book	$Gjmb$

Predicate logic treats both verbs and nouns as standing for properties of objects, even though their syntax and communicative function is different in natural language. The predicate logical form of "John walks" uses a predicate letter and a single constant. The form of "John is a boy" also uses a predicate letter with a constant: Bj.

These examples demonstrate the variety of predication in natural language: intransitive verbs like 'Walk" take one object, transitive verbs like "see" take two, verbs like "give" even take three. The same variety occurs in mathematics as we will see a bit later, and it is essential to predicate logic: atomic statements express basic properties of one or more objects together. In the history of logic, this is a relatively late insight. The theory of syllogistics describes only properties of single objects, not relations between two or more objects.

Exercise 4.1 The hold of the syllogistic of our preceding chapter, and its emphasis on "unary" properties of single objects has been so strong that many people have tried to reduce binary predicates to unary ones. One frequent proposal has been to read, say, "x is smaller than y" as "x is small and y is not small". Discuss this, and show why it is not adequate. Does it help here to make the property "small" context-dependent: "small compared to..."?

Translation key Note that in writing predicate logical translations, one has to choose a "key" that matches natural language expressions with corresponding logical letters. And then stick to it. For mnemonic purposes, we often choose a capital letter for a predicate as close to the natural language expression as we can (e.g., B for "boy"). Technically, in the logical notation, we should indicate the exact number of object places that the predicate takes ("B has one object place"), but we drop this information when it is clear from context. The object places of predicates are also called *argument places*. If a predicate takes more than one argument, the key should say in which order you read the arguments. E.g., our key here is that Sjm says that John sees Mary, not that Mary sees John. The latter would be Smj.

Predicates in language and mathematics Let us discuss predicates a bit further, since their variety is so important to predicate logic. In mathematics, 2-place predicates are most frequent. Common examples are $=$ ('is equal to'), $<$ ('is smaller than'), $\in$ ('is an element of'). It is usual to write these predicates in between their arguments: $2 < 3$. (We will say more about the expressive possibilities of the predicate "$=$" on page 4-41.) Occasionally, we also have 3-place predicates. An example from geometry is "x lies between y and z", an example from natural language is the word "give" (with a giver, an object, and a recipient).

informal mathematics	*logical/mathematical formula*
Two is smaller than three	$2 < 3$
x is smaller than three	$x < 3$
x is even (i.e., 2 divides x)	$2\|x$
Point p lies between q and r	$Bpqr$

In the special case of talking about mathematics there are standard names for objects and relations. In $x < 3$, the term "3" is a constant that names a particular natural number, and "$<$" is a standard name for a specific relation. The notation $x|y$ expresses that x is a divisor of y, i.e., that division of y by x leaves no remainder. Natural language also has special names for distinguished objects, such as "Alexander the Great", "Indira Ghandi", or "The Great Wall".

Note that betweenness is not a conjunction of "x lies between y" and "x lies between z": that would be nonsense. However, an abbreviation that is often used in mathematics is $x < y < z$, to express that the number y is in between x and z. This is not an example of real 3-place predicate. Rather, it is an abbreviation of $x < y \wedge y < z$. In this special case, when an order is 'linear', betweenness does reduce to a conjunction after all.

Exercise 4.2 Express $\neg(x < y < z)$ in terms of the binary predicate $<$ and propositional connectives, using the fact that $x < y < z$ is an abbreviation of $x < y \wedge y < z$.

The standard in predicate logic is to write the predicate first, then the objects. The exceptions to this rule are the names for binary relations in mathematics: $<$ for less than, $>$ for greater than, and so on. The general rule is for uniformity, and it takes getting used to. Many natural languages put predicates in the middle (English, French, but also the informal language of mathematics), but other languages put them first, or last. Dutch and German are interesting, since they put predicates in the middle in main clauses ("Jan zag Marie"), but shift the predicate to the end in subordinate clauses ("Ik hoorde dat Jan Marie zag").

Referring to objects: pronouns and variables We already saw how proper names like "John" or "Mary" refer to specific objects, for which we wrote constants like a, b. But both natural language and mathematics use 'variable names' as well, that stand for different objects in different contexts. Pronouns in language work like this: "John sees her" (Sjx) refers to some contextually determined female "her", and $x < 2$ expresses that some contextually determined number x is smaller than 2. Think of a geometry textbook where x is introduced as the side of a triangle with two other sides of length 1. The famous author Italo Calvino once wittily called pronouns "the lice of thought" [Cal88]. But are they just a nuisance? To the contrary, what pronouns do is provide coherence in what you say, by referring back to the same individual in the right places. That is also exactly what mathematical variables do. So we get analogies between pronouns in natural language and contextually determined variables in mathematics, like:

John sees her	Sjx
He sees her	Syx
This is less than that	$x < y$
He sees himself	Sxx

Note that whether $x < y$ is true totally depends on specifying x and y. It is true for $x = 2$ and $y = 5$ but false for $x = 5$ and $y = 2$. 'This' and 'that' in natural language are demonstrative pronouns for pointing at things. Mathematicians use variables as pointers. Instead of "suppose this is some number greater than one" they say "suppose x is a number greater than one". Next, they use x to refer to this number.

Adding on propositional logic Propositional operators can be added in the obvious way to the preceding statements, with the same function as before:

John does not see Mary	$\neg Sjm$
Three is not less than two	$\neg 3 < 2$, or abbreviated: $3 \not< 2$.
John sees Mary or Paula	$Sjm \vee Sjp$
Three is less than three or three is less than four	$3 < 3 \vee 3 < 4$.
x is odd (i.e., two does not divide x)	$\neg(2\|x)$
If John sees Mary, he is happy	$Sjm \rightarrow Hj$

As a small detail in style, in the last example, natural language uses a pronoun ("he is happy"), while the logical translation does not use a variable but a second occurrence of the constant j to refer to the same object as before. Logic is more precise, natural language is more flexible. The reuse of the constant j rules out any possibility of misunderstanding, while the correct interpretation of "he" still depends on context.

Exercise 4.3 Translate the following sentences into predicate logical formulas:

(1) If John loves Mary, then Mary loves John too.

(2) John and Mary love each other.

(3) John and Mary don't love each other.

(4) If John and Peter love Mary then neither Peter nor Mary loves John.

Exercise 4.4 Rephrase "$x \leq y \wedge y \leq z$" in predicate logic, using binary relations $<$ for "less than" and $=$ for "equal".

Exercise 4.5 Rephrase "$\neg(x \leq y \wedge y \leq z)$" in predicate logic, using binary relations $<$ for "less than" and $=$ for "equal".

A first glimpse of quantifiers Now we move to the most important new operators, quantifiers which say that objects exist without naming them explicitly:

Someone walks	$\exists x Wx$
Some boy walks	$\exists x(Bx \wedge Wx)$
A boy walks	$\exists x(Bx \wedge Wx)$
John sees a girl	$\exists x(Gx \wedge Sjx)$
A girl sees John	$\exists x(Gx \wedge Sxj)$
A girl sees herself	$\exists x(Gx \wedge Sxx)$

Here you can see what variables do: they keep track of which object does what when that object occurs several times in the sentence. With long sentences or long texts, a systematic mechanism like this becomes crucial in keeping reasoning straight.

Here are some examples with universal quantifiers:

Everyone walks	$\forall x Wx$
Every boy walks	$\forall x(Bx \rightarrow Wx)$
Every girl sees Mary	$\forall x(Gx \rightarrow Sxm)$

Note that "Some boy walks" is translated with a conjunction symbol, and "Every boy walks" is translated using an implication symbol. The following exercise explains why.

Exercise 4.6 Let B be the predicate for "boy" and W the predicate for "walk".

(1) What does $\forall x(Bx \wedge Wx)$ express?

(2) And what does $\exists x(Bx \rightarrow Wx)$ express?

At first, you may find this predicate-logical formulation somewhat strange. But give it some more thought, and you may come to see that the predicate-logical formulas really get to the heart of the meaning and structure of objects.

Remark on natural language and logic Our examples show similarities between natural language and logic, but also a bit of friction. It may seem that "John is a boy" contains an existential quantifier ("there is a boy"), but this appearance is misleading. This has been much discussed: already ancient Greek logicians felt that their own language has a redundancy here: the indefinite article "a" is redundant. Indeed, many natural languages do not put an "a" in this position. An example from Latin: *"puer est"* ("he is a child"). But such languages also do not put an "a" in positions where there is a real existential quantifier. Another example from Latin: *"puer natus est"* ("a child is born"). Likewise, the ancient Greek logicians already observed that the "is" of predication in "John is a boy" does not seem to do any real work, and again, might just be an accident of language. As one can imagine, debates about how natural language structure relates to logical form are far from over — and they remain a powerful inspiration for new research.

Quantifier combinations The advantages of the austere notation of predicate logic quickly become clearer when we combine quantifiers. This creates patterns that are essentially more sophisticated than those in our previous chapter on syllogistic reasoning. Combined patterns occur widely in science and natural language, but predicate logic makes them especially perspicuous:

Everyone sees someone	$\forall x \exists y Sxy$
Someone sees everyone	$\exists x \forall y Sxy$
Everyone is seen by someone	$\forall x \exists y Syx$
Someone is seen by everyone	$\exists x \forall y Syx$

You can again think of mathematical examples just as easily:"every number has a greater number", "some number has no smaller number", "some sets have no elements", etc.

The domain of quantifiers Quantifiers in natural and formal languages come with an additional feature, that we left implicit so far. The quantifiers range over all objects in some given set of relevant objects, the *domain of discourse*. In the above natural language examples, the domain only contains human beings, perhaps even just a small set of them — in mathematical examples, it can be numbers or geometrical figures. But in principle, every set of objects, large or small, can be a domain of discourse. We will make this feature more explicit later when we talk about the semantics of predicate logic.

Technically, the restriction to domains of discourse is sometimes reflected in how we "restrict" a quantifier to some relevant subset, indicated by a unary predicate. For instance, in the obvious translations of syllogistic statements like "All A are B", the formula $\forall x(\mathbf{A}x \rightarrow Bx)$ has its universal quantifier restricted to the predicate A, and likewise, the existential quantifier for "Some A are B" is restricted as follows: $\exists x(\mathbf{A}x \wedge Bx)$.

This completes our first tour of predicate logic. The next section will help you in much more detail with finding logical forms for natural language sentences.

4.2 Practising Translations

In a logic course, going from sentences to formulas is taught as a sort of art allowing you to see the structure of ordinary assertions, and doing inference with them. And this art also makes sense for our other extreme of mathematics: mathematicians also speak natural language, be it with many special notations, and they have never switched completely to using only formulas. The modern area of "natural language processing" has developed the translation process also into a sort of science, where computers actually translate given natural language sentences into logical representations. In what follows, we go through some detailed examples that may help you develop a methodical style of translating.

If you feel that you get the picture by now, you can skip this section.

Translating simple syllogistic sentences Here is how the syllogistic statements of the preceding chapter can be expressed in predicate logic:

$$\begin{array}{rc} \text{All A are B} & \forall x(Ax \to Bx) \\ \text{No A are B} & \neg\exists x(Ax \wedge Bx) \\ \text{Some A are B} & \exists x(Ax \wedge Bx) \\ \text{Not all A are B} & \neg\forall x(Ax \to Bx) \end{array}$$

Exercise 4.7 Give the predicate logical formulas for the following syllogistic statements:

(1) No B are C.

(2) Some A are C.

But the real power of predicate logic shows only in combinations.

Translating sentences with multiple quantifiers We can also use these patterns for finding translations for sentences involving double quantifications, as in the following example:

$$\text{Every boy loves a girl.} \tag{4.1}$$

To begin with we translate the pattern for "All B are ...":

$$\forall x\,(Bx \to \varphi(x)) \tag{4.2}$$

Here B is a unary predicate to represent 'boy', and $\varphi(x)$ stands for the property that we want to assign to all the boys x: "x loves a girl". This part of the sentence is in fact something of the form "Some C are D", where C represents the class of girls and D the class of those objects which are loved by x. The predicate logical translation of $\varphi(x)$ therefore looks as follows:

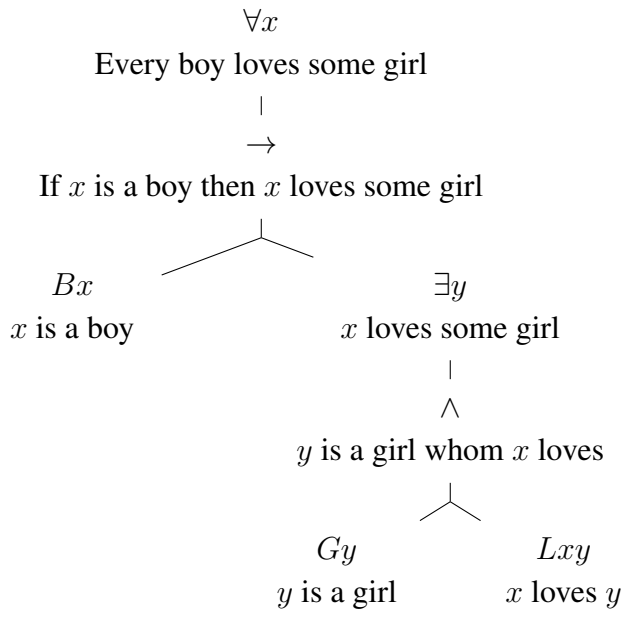
$$\exists y\,(Gy \wedge Lxy) \tag{4.3}$$

with Gy for "y is a girl" and Lxy for "x loves y". Substitution of this translation for $\varphi(x)$ in (4.2) gives us the full predicate logical translation of (4.1):

$$\forall x\,(Bx \to \exists y\,(Gy \wedge Lxy)) \tag{4.4}$$

In the following figure, a parse tree of the translation is given which shows the compositional structure of predicate logic. Every subformula corresponds to a sentence in natural

language:



(4.5)

This way of constructing translations can be of great help in finding predicate logical formulas for natural language sentences.

In ordinary language we are used to writing from left to right, but in predicate logic the order of putting together connectives and quantifiers is often non-linear. The following longer example illustrates this.

No girl who loves a boy is not loved by some boy. (4.6)

Although this sentence looks quite complicated, its surface structure coincides with the syllogistic form "No A are B", and so our first translation step is:

$$\neg\exists x\,(\varphi(x) \wedge \psi(x)) \tag{4.7}$$

where $\varphi(x)$ are those x who are girls who love some boy, and $\psi(x)$ represents the class of those x who are loved by no boy. The first part, $\varphi(x)$, is a conjunction of two properties: x is a girl *and* x loves a boy. This is just a small step towards a complete translation:

$$\neg\exists x\,((Gx \wedge \varphi_1(x)) \wedge \psi(x)) \tag{4.8}$$

with $\varphi_1(x)$ representing "x loves a boy". This part can be translated into:

$$\exists y\,(By \wedge Lxy) \tag{4.9}$$

Substitution of this result for $\varphi_1(x)$ in (4.8) gives us the following intermediate result:

$$\neg\exists x\,((Gx \wedge \exists y\,(By \wedge Lxy)) \wedge \psi(x)) \tag{4.10}$$

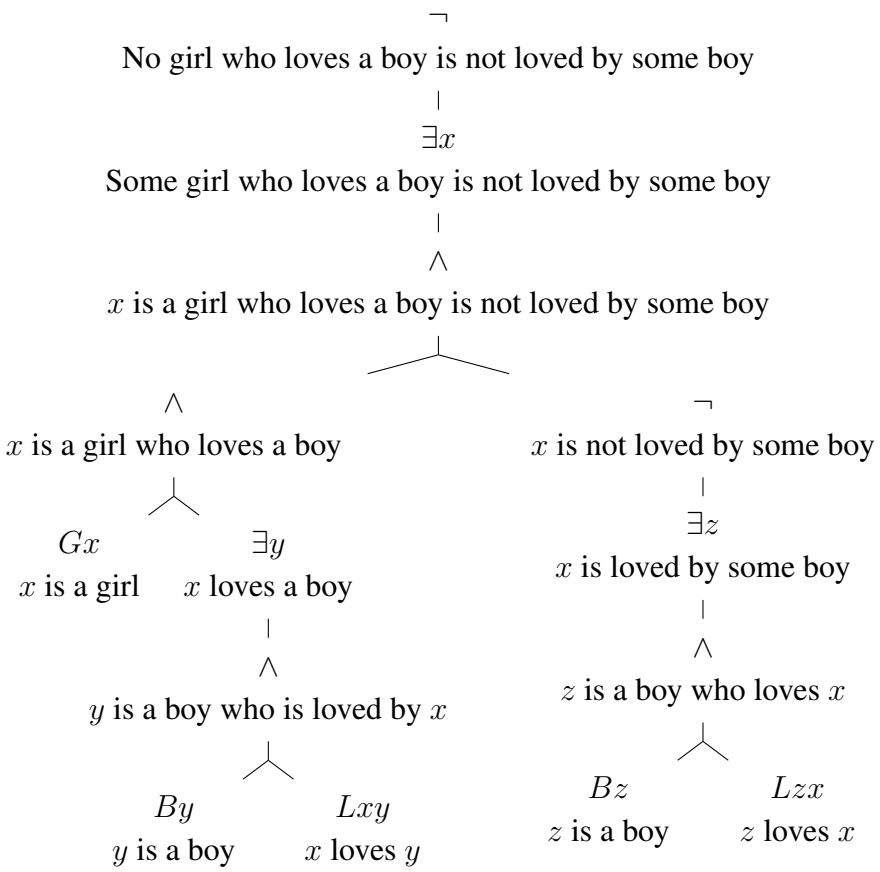
The subformula $\psi(x)$ represents "No boy loves x", which can be translated into:

$$\neg\exists z\,(Bz \wedge Lzx) \tag{4.11}$$

This yields the final result:

$$\neg\exists x\,((Gx \wedge \exists y\,(By \wedge Lxy)) \wedge \neg\exists z\,(Bz \wedge Lzx)) \tag{4.12}$$

Below a complete composition tree is given for this translation. Again, every subformula can be paraphrased in natural language:



(4.13)

Iterating quantifiers: twice, and more Two-quantifier combinations occur in natural language as we have just seen, and they are also very common in mathematics. The same logical form that expressed 'Everyone sees someone' is also that for a statement like 'Every number has a larger number'. And the above form for 'Some girl sees every boy' is also that for 'There is an odd number that divides every even number' (namely, the number 1).

Can there be still higher nestings of quantifiers? Yes, indeed. For instance, three quantifiers are involved in the famous saying that "You can fool some people some of the time,

and you can fool some people all of the time, but you cannot fool all people all of the time". To see that this really involves three quantifiers, observe that the "you" must be read as "someone". The translation of "Someone can fool some people some of the time" (with P for "person", T for "instant of time", F for "fooling"):

$$\exists x(Px \wedge \exists y(Py \wedge \exists z(Tz \wedge Fxyz))).$$

And for "Someone cannot fool all people all of the time":

$$\neg\exists x(Px \wedge \forall y(Py \rightarrow \forall z(Tz \rightarrow Fxyz))).$$

Likewise, three-quantifier combinations occur in mathematics. A typical example is the definition of 'continuity' of a function f in a point x:

> For every number r, there is a number s such that for all y with $|x - y| < s$: $|f(x) - f(y)| < r$.

Nestings of four quantifiers are rare, they get hard for humans to understand.

Exercise 4.8 Assume the domain of discourse to be all human beings. Translate the following sentences into predicate logic:

(1) Augustus is not loved by everyone. (Use a for Augustus, L for love.)

(2) Augustus and Livia respect each other. (Use a for Augustus, l for Livia and R for respect.)

(3) Livia respects everyone who loves Augustus.

Exercise 4.9 Assume the domain of discourse is all animals. Translate: *Some birds do not fly.* (Use B for being a bird and F for being able to fly.)

Exercise 4.10 For each of the following, specify an appropriate domain of discourse, specify a key, and translate into predicate logic. (Note: you have to understand what a sentence means before you can attempt to translate it.)

(1) Dogs that bark do not bite.

(2) All that glitters is not gold.

(3) Friends of Michelle's friends are her friends.

(4) There is a least natural number.

(5) There is no largest prime number.

Exercise 4.11 Translate the following sentences into predicate logical formulas. Assume the domain of discourse is human beings.

(1) Every boy loves Mary.

(2) Not all girls love themselves.

(3) No boy or girl loves Peter.

(4) Peter loves some girl that loves John.

Exercise 4.12 For each sentence from the previous exercise, draw a picture that makes it true.

Exercise 4.13 Translate the following sentences into predicate logical formulas. Assume the domain of discourse is human beings.

(1) Some boy doesn't love all girls.

(2) Every boy who loves a girl is also loved by some girl.

(3) Every girl who loves all boys does not love every girl.

(4) No girl who does not love a boy loves a girl who loves a boy.

4.3 Reasoning Patterns with Quantifiers

Let us now turn to the kind of reasoning that predicate logic brings to light. As with earlier systems, the basis here is your intuitive understanding of reasoning, in this case, with quantifiers. The formal system then makes this more precise and systematic. In this section, we look at some valid inferences, without formal analysis: we are appealing to your intuitions.

Monadic Predicate Logic Before unleashing the full power of quantifiers in predicate logic, we first consider a more modest stepping stone.

The language of the Syllogism (Chapter 3) is a small fragment of predicate logic that imposes a restriction on the form of the predicates that are allowed: they have to be "unary properties", with atomic statements involving one object only. This special system with only 1-place predicates (unary properties of objects) is called *monadic predicate logic*. This restriction on predicate form curbs the power of quantification considerably, but it also makes it easier to understand. Let us see how syllogistic reasoning can be expressed in monadic predicate logic.

Consider the syllogism with premises "All A are B" and "All B are C" and conclusion 'All A are C". It corresponds to the valid predicate-logical inference

$$\forall x(Ax \to Bx), \forall x(Bx \to Cx) \textit{ imply } \forall x(Cx \to Ax)$$

by using a small variation of one of the methods of Chapter 3.

Of course, you have already learnt the Venn Diagram method that tests such inferences for validity or invalidity. More in terms of predicate logic, here are some further patterns. Syllogistic theory has the following equivalences:

- *Not all A are B* has the same meaning as *Some A are not B*.
- *All A are not B* has the same meaning is *there are no A that are also B*.

The predicate logical versions of these equivalences give important information about the interaction between quantification and negation:

- $\neg\forall x(Ax \rightarrow Bx)$ is equivalent to $\exists x\neg(Ax \rightarrow Bx)$, which is in turn equivalent to $\exists x(Ax \wedge \neg Bx)$,
- $\forall x(Ax \rightarrow \neg Bx)$ is equivalent to $\neg\exists x\neg(Ax \rightarrow \neg Bx)$, which is in turn equivalent to $\neg\exists x(Ax \wedge Bx)$.

From this we can distill some important general quantification principles:

- $\neg\forall x\varphi$ is equivalent to $\exists x\neg\varphi$,
- $\neg\exists x\varphi$ is equivalent to $\forall x\neg\varphi$.

Reflecting on these principles a bit further, we see that negation allows us to express one quantifier in terms of the other, as follows:

- $\forall x\varphi$ is equivalent to $\neg\exists x\neg\varphi$,
- $\exists x\varphi$ is equivalent to $\neg\forall x\neg\varphi$.

Exercise 4.14 Translate the following syllogistic statements into predicate logic, without using existential quantifiers:

(1) Some A are B.

(2) Some A are not B.

(3) No A are B.

Exercise 4.15 Translate the following syllogistic statements into predicate logic, without using universal quantifiers:

(1) All A are B.

(2) All A are non-B.

(3) No A are B.

Actually, monadic predicate logic also has basic inferences that are not syllogistic in nature. Here is a key example:

$\forall$ distributes over $\wedge$.

On the other hand, $\forall$ does not distribute over $\vee$. Here is a simple counterexample. The Greeks held the view that all people are either Greeks or Barbarians. So, with *all people* as the domain of discourse:

$$\forall x(Gx \vee Bx).$$

But they did certainly not hold the view that either all people are Greek or all people are Barbarians. For they knew that the following was *false*:

$$\forall xGx \vee \forall xBx.$$

Exercise 4.16 What are the analogous observations for $\exists$, $\wedge$ and $\vee$?

Syllogistic reasoning is still very simple, and indeed, validity in monadic predicate logic, which has unary predicates only, is still *decidable*: there are mechanical methods for testing whether given inferences are valid, as we have seen in Chapter 3. This is no longer true for predicate logic as a whole, but we will come to that point only later.

Logic and natural language once more Modern logicians view the syllogistic as a tiny (monadic) fragment of the much richer full natural language. From that point of view, traditional logic was extremely weak, and its success in history becomes hard to understand. But this is an essential misunderstanding of how a logical system functions. Syllogistic forms are used for *analysis* of given sentences, and they provide a sort of top-level logical form, while the predicates A, B, etc. may stand for large chunks of natural language text of potentially very high (non-syllogistic!) complexity. For instance, here is a surface "All A are B" form: "All prisoners who have inflicted much damage on several people (A) fall under the provisions of most criminal codes invented by different cultures (B)".

In this sense you should not misunderstand the art of translation that we have shown you. Outside the realm of our toy examples, it is usually hopeless to translate a complete natural language sentence into a logical formula. The point is rather that you formalize part of the outer structure of the sentences involved, say, enough to see whether a claimed inference is valid or not. Leaving large internal chunks of the sentences unformalized is not a problem in such a case: it is rather the sign of a certain sense for taste and elegance.

Reasoning with binary and higher predicates, and with quantifiers Valid reasoning in predicate logic extends far beyond syllogistic forms. The following is a famous 19th century example of a non-syllogistic valid inference involving binary relations:

"All horses are animal", therefore: "All horse tails are animal tails"

Using a binary predicate P for "possess" or "have", we can translate this as follows:

$$\frac{\forall x(Hx \to Ax)}{\forall y((Ty \wedge \exists x(Hx \wedge Pxy)) \to \exists z(Az \wedge Pzy))}$$

This is one instance of a general phenomenon: "monotonicity inferences" with predicate replacement of the sort discussed in the preceding chapter work in much greater generality than just the syllogistic. Predicate logic tells you how.

Another natural set of inferences has to do with iterated quantifiers. First, here are some rather trivial, but nevertheless valid inferences:

$\forall x \forall y \varphi \leftrightarrow \forall y \forall x \varphi$ is valid.

Much more interesting, of course, are other combinations. For a start, it is easy to see that a quantifier combination $\forall x \exists y$ does not imply $\exists y \forall x$: everyone has a mother, but no one is a mother of everyone. And even $\forall x \exists y Rxy$ does not imply the same shape with variables permuted: $\forall x \exists y Ryx$. Everyone has a parent, but not everyone has a child. Here is about the only interesting valid inference between 2-quantifier combinations:

$\exists x \forall y \varphi$ implies $\forall y \exists x \varphi$.

You may formulate for yourself why this is intuitively (and really) valid.

Exercise 4.17 Determine validity or invalidity of all the remaining possible implications between repeated quantifiers.

This ends our list of intuitively valid principles of reasoning with quantifiers.

Warning: the role of compositionality We do not want to leave you with the wrong impression. From our progressive list, you might think that we have now classified 2-quantifier inferences, then we should go on to 3-quantifier ones, and so on. This was indeed how some medieval logicians saw the task ahead for logic. But this is mistaken. We do not have to go up the hierarchy that seemed in the making. There is a complete proof system for predicate logic whose rules just tell us explicitly what single quantifiers do. All the valid behaviour of complex quantifier combinations then follows automatically by finding the right *combinations* of proof steps for single quantifiers.

4.4 Formulas, Situations and Pictures

By now, it will have become clear to you that predicate logic is a very general language for talking about situations where objects have certain properties or are in certain relations. But what that really means is that these situations are crucial to understanding truth or falsity of formulas. Therefore, let us talk a bit more about what they are, intuitively, and how they connect up with formulas.

You can also see this point as follows. The semantics of a language ties the syntactic expressions of that language to real situations that language users talk about, say when communicating information to each other. Thus semantics has two independent pillars: language, and reality: the situations which that language can in principle describe.

Scenes from daily life Here is a simple example from everyday life:

Suppose we use B for the property of being a bicycle, M for the property of being a man, and R for the relation of riding, then we can say that the following formula describes the following salient fact about the situation in the picture:

$$\exists x \exists y (Mx \wedge By \wedge Rxy).$$

Of course, you can also write statements about the trees.

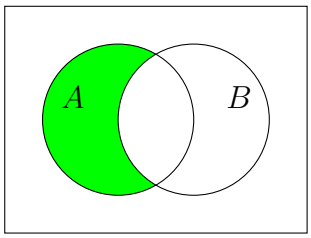
Exercise 4.18 Consider the following picture:

Use G for the property of being a girl, H for the property of being a hat, and W for the relation of wearing (so that Wxy expresses that x is wearing y). Now do the following:

(1) Give a predicate logical formula that is *true* for the situation in the picture.

(2) Give a predicate logical formula that is *false* for the situation in the picture.

Venn diagrams The Venn diagrams that you know from our chapter on syllogistic reasoning are abstract pictures of situations where objects have certain properties. Look at the Venn picture of the set of things that are in A but not in B:



(4.14)

Here is one relevant predicate logical formula:

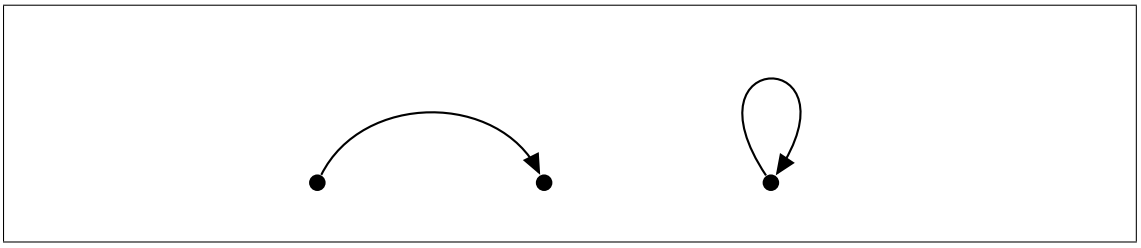
$$Ax \wedge \neg Bx. \tag{4.15}$$

The green area in the picture corresponds to the set of individuals x that make formula 4.15 true. Here, the formula (4.15) contains a *free variable* x, that is, x is *not bound* by a quantifier. This serves as a place-holder where different objects can be put to make true or false statements, and hence this formula describes a property of objects. Of course, this is exactly how mathematicians use variables as well: given the situation of the real numbers, the formula $x^2 < x$ defines a particular subset, namely the real numbers from 0 up to, but not including 1.

Graphs for representing relations A widely used concrete picture for a situation is a *graph*: a set of points connected by lines (often called an "undirected graph") or by directed arrows (a "directed graph"). Graphs are used everywhere in science, from pure mathematics to social networks, and also in daily life: the NS railway map of The Netherlands is a graph of cities and railway connections. In fact, they have already been used in this course itself! *Trees* are a special kind of graph with nodes and connections from nodes to their children, and you have already used trees to represent logical formulas.

Graphs are one of the most useful mathematical structures all around, being abstract yet simple to grasp. They are often used to demonstrate points about logic.

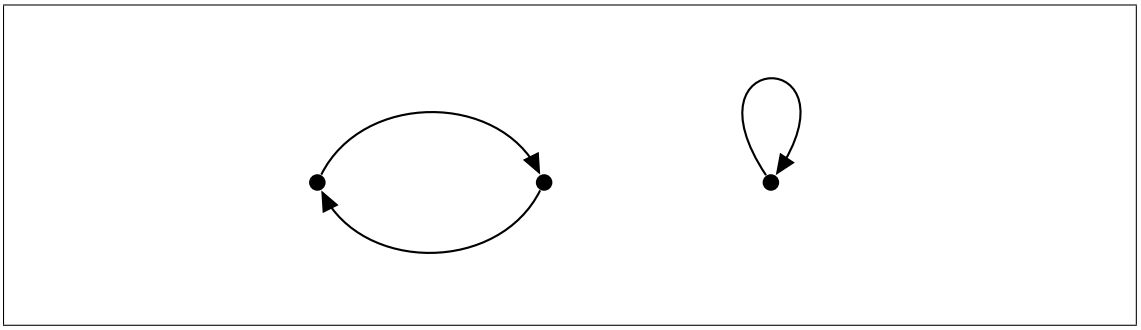
Here is a picture of a directed graph: a set of nodes, linked by arrows called "edges":



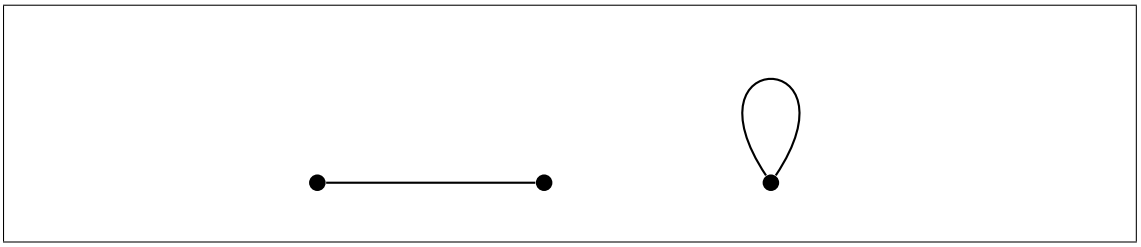
Let us take one binary predicate R for expressing that two objects in the picture are linked by an edge, with the arrow pointing from the first to the second object. Then it is easy to see that the following formulas are true:

$$\exists x \exists y Rxy$$
$$\exists x Rxx$$
$$\neg \forall x \exists y Rxy$$

Now let us change the picture by adding one more edge, to make the relation *symmetric*:



Effectively, this changes the picture into an *undirected* graph: since every arrow can be reversed, there is no need anymore to indicate the direction of the edges, and the picture can be simplified, as follows:



The formula $\forall x \forall y(Rxy \rightarrow Ryx)$, is not just true here, but is true in all undirected graphs: it expresses that the link relation is *symmetric*.

Symmetry is one of many important properties of binary relations that can be defined in predicate logic. We will see a few more in a moment.

Exercise 4.19 The formula

$$\forall x \forall z(\exists y(Rxy \wedge Ryz) \rightarrow Rxz)$$

expresses *transitivity* of the relation R. Which of the following relations are transitive:

(1) being an ancestor of ... on the set of human beings,

(2) being a parent of ... on the set of human beings,

(3) the 'less than' relation $<$ on the natural numbers,

Example 4.20 The binary relation in the last picture above is *not* transitive. Here is why not. Consider the left and the middle point in the picture. There is a link from left to middle, and there is a link from middle to left. This is so because, as we have just seen, the relation in the picture is symmetric. Now transitivity says that if you can get somewhere in two steps by following a relation, then you can also get there in a single step in that relation. Thus, transitivity demands that you can go from the left point to the left point in a single step. This is not the case, so the relation is not transitive.

Exercise 4.21 A relation R is called *linear* if the following holds:

$$\forall x \forall y(Rxy \vee x = y \vee Ryx).$$

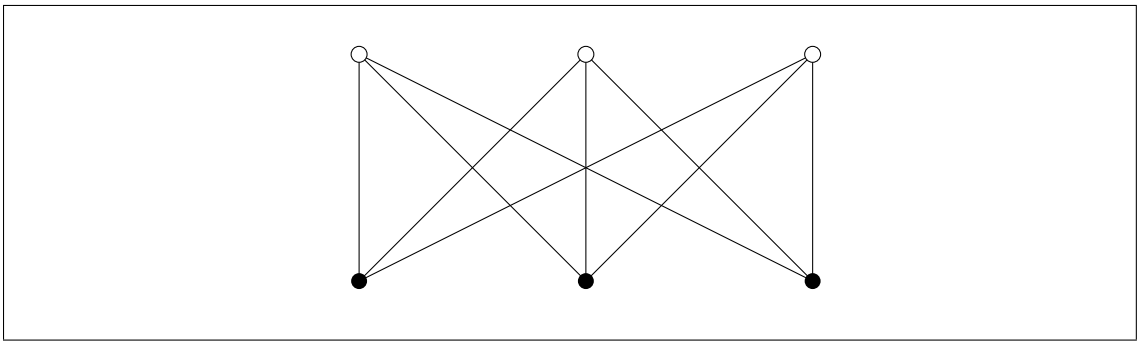
Clearly, the relation in the last picture above is not linear, for there is no link between the middle point and the point on the right in the picture. Which of the following relations are linear:

(1) being an ancestor of ... on the set of human beings,

(2) being a parent of ... on the set of human beings,

(3) the 'less than' relation $<$ on the natural numbers,

The next graph has a distinction between two kinds of nodes. This is something we can talk about with an additional 1-place predicate:



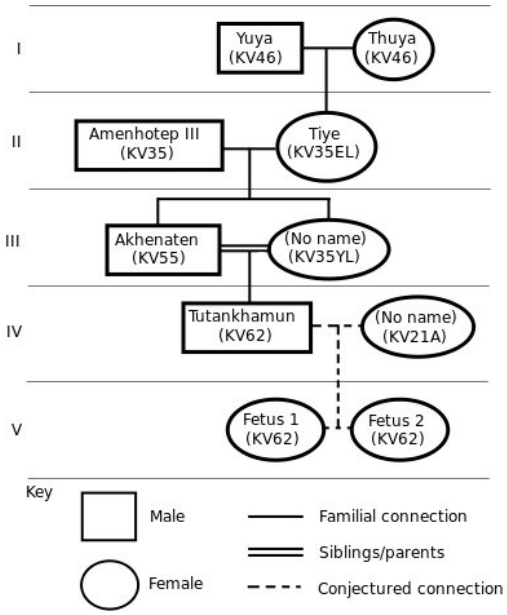
Let us say that the solid dots • have the property P, and the open dots ∘ lack that property. The true statement "All solid dots are linked to at least one open dot" can now be expressed as follows:

$$\forall x(Px \to \exists y(\neg Py \wedge Rxy)).$$

And here is how we express that links are only between solid and open dots:

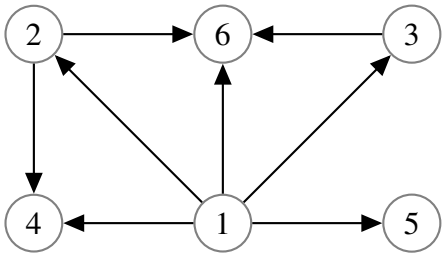
$$\forall x \forall y(Rxy \to (Px \leftrightarrow \neg Py)).$$

Most useful graphs in logic are directed, with arrows making links. A very nice example are family relations, as represented in genealogical *trees*. Here is the family tree of the famous Egyptian pharao Tutankhamun, as revealed by DNA research in 2010:



Exercise 4.22 Using relation symbols P for "parent of", and M for male, give a predicate logical formula that expresses the surprising fact (well, not so surprising, if you know something about ancient Egyptian royal families) that came to light about the lineage of Tutankhamun: *The parents of Tutankhamun were brother and sister.*

Directed graphs visualize relations of many kinds. A concrete example is *divisibility* on the natural numbers $\{1, 2, 3, 4, 5, 6\}$:



(4.16)

The circles around the numbers represent reflexive arrows: every number divides itself: $\forall x\,(x|x)$, where $|$ is used as an infix symbol for the relation of divisibility. Another basic property of the divisibility relation is *transitivity*:

$$\forall x \forall y \forall z\,((x|y \wedge y|z) \rightarrow x|z).$$

This says that if one number divides a second, and that second a third, then the first number divides the third. Many relations are transitive: think of being older than, being as old as, being no older than (think about this!), being an ancestor of. Another general feature of divisibility is its *anti-symmetry*:

$$\forall x \forall y((x|y \wedge y|x) \rightarrow x = y)$$

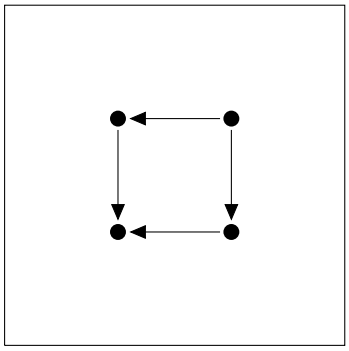
This, too, holds for many relations: though not for being "as old as": why not?

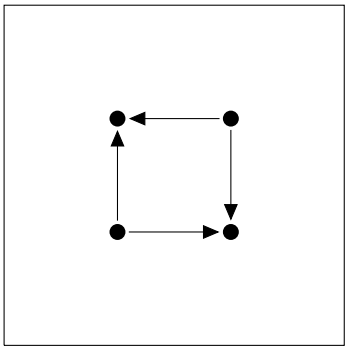
Example 4.23 We can use the predicate for "divides" to give a predicate logical definition of being a prime number, as follows. A prime number is a natural number with the property that it has no proper divisors greater than 1. A proper divisor of a number x is a number y that is smaller than x and that divides y. In predicate logic: y is a proper divisor of x greater than 1 if $1 < y \wedge y < x \wedge y|x$. Therefore, the following formula gives a definition of being prime:

$$P(x) \leftrightarrow \neg\exists y(1 < y \wedge y < x \wedge y|x).$$

You will encounter many more graphs later on in this course. Chapter 5 on knowledge and communication will have many undirected ones, Chapter 6 on action has directed graphs. We now state a few more exercises, since graphs concretely visualize much of what we are going to say more abstractly in the following sections.

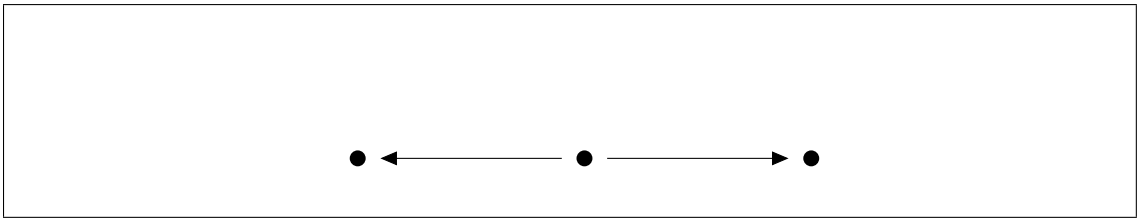
Exercise 4.24 Predicate logic can be used to tell two graphs apart: find a formula that is true in the one but not in the other. Consider the following two pictures:





Give a formula, with R for the relation, that is *true* on the left and *false* on the right.

Exercise 4.25 ♡ When a formula is false in a graph, we may want to *change* that graph in a minimal manner to make the formula true after all. Think of an engineer changing a blueprint to meet a given specification. Consider the following graph:



The formula $\exists x \forall y Rxy$ is false in the graph if R is interpreted as the $\rightarrow$ relation. Make the formula true by adding a single $\rightarrow$ link to the graph.

Exercise 4.26 ♠ Here is a harder problem. Suppose that we are talking about finite directed graphs whose ordering is reflexive and *connected*, where the latter property means that for any two points, an arrow runs from at least one to the other. (How would you write this as a formula?) Many communication networks are like this. Now call a point in the graph a "Great Communicator" if it can reach every point in the graph by one, or at most two successive arrows. (How would you write this in a formula?) Now the real question:

> Show that every finite connected graph has a "Great Communicator".

You may want to draw some pictures first to get a feeling for what these graphs are like. While we are at it, here is one more question. The stated inference does *not* hold for infinite graphs: can you give a counter-example?

We conclude with a few general observations on the link between predicate-logical formulas and situations (from numbers to pharao's) that we have now demonstrated in many concrete settings.

"The missing link": semantic interpretation Assigning meaning to a language really involves *three ingredients*, not just the two we discussed so far. We already had the sentences of the language, a syntactic code. Then we looked at the situations described, consisting of objects with structure: properties, relations, and the like. But the syntactic code only acquires meaning by *connecting* it with structures, a process of "interpretation". If you receive a letter in a language that you do not know, you just have the code. If you see a newspaper article in a language that you do not know, but with an inserted picture of Mount Etna, you may know the situation (yesterday's spectacular eruption) and the code, but you still do not know what the message says, since you do not know its interpretation. This three-way scheme gives language an amazing versatility. One piece of syntactic code can talk about many situations, and even about one situation under different interpretations, one situation can be described in many languages, and so on.

These ideas apply also to the language of predicate logic. As we have seen, it describes "universes of discourse"' consisting of objects, but we need a link. In the above, this link was often given informally as a "key" mapping predicate-logical symbols to corresponding structural components of the relevant situation. How does this linkage work in general?

In propositional logic, the link was the *valuation* mapping proposition letters to truth values. But this will no longer do. For checking whether a statement saying that a certain object has a certain property, or that certain objects are in a certain relation is true we need something more refined. Instead of just saying that "John is boy" is assigned the value *true*, we now need an interpretation for "John" and an interpretation for "being a boy". Here is a list, to give you a first impression of what we need:

(1) Individual constants denote concrete objects,much like proper names.

(2) Unary predicate letters denote concrete properties of objects, and likewise, binary predicate letters concrete binary relations, and so on. (We can think of these assigned predicates in terms of sets of objects or of ordered tuples and so on, but this is just a particular mathematical implementation, not the essence of the matter.)

(3) Variables will not denote fixed objects, but they can run through objects: something for which we will use a special device later on called an "assignment".

(4) Logical expressions like Boolean connectives and quantifiers are given their standard meaning that we have already seen at work throughout this text.

Interpreting complex sentences in stages Mapping the basic alphabet of the language to corresponding items in a situation allows us to interpret atomic sentences saying that one or more objects stand in some relation. What about more complex sentences formed with perhaps lots of logical operations, a set containing infinitely many expressions of extremely high syntactic complexity?

One of the many amazing things about predicate logic, and indeed, one of the major discoveries by its 19th century founding fathers, is that we do not have to worry about this. As we will see in more formal detail soon, formulas with nested quantifiers are constructed systematically by *stepwise syntax rules* introducing one symbol at a time. Next, these rules can be *interpreted semantically*, and then meanings for expressions of arbitrary complexity just arise automatically by calling on the meaning of single propositional operators and quantifiers as many times as required.

This process of stepwise interpretation from components is called *compositionality*, and it is a major design principle, not just in logic, but also, for instance, in programming languages or algebraic formalisms in mathematics. Compositionality is also important to philosophers, since it seems to provide an explanation for what is after all a very mysterious phenomenon. On the basis of only a finite amount of information (sample sentences from your parents, perhaps some grammatical rules in school) competent speakers of a language understand a potential infinity of new sentences when confronted with them, even when they have never seen them before.

4.5 Syntax of Predicate Logic

The time has come to confront the formal details of how predicate logic works. You have now seen informally how the language of predicate logic works in a number of settings, and how it can describe various structures from mathematics to our daily world. Next, you may want to see how it works for the same purposes that you have seen in earlier chapters, such as information update and, in particular, inference. But before we do that, it is time to sharpen up things, and say exactly what we mean by the language and semantics of predicate logic.

The first step is an investment in 'formal grammar'. It looks a bit technical, but still a far cry from the complexities of a real natural language like English. The following grammar is a simple model of basics of the grammar of many languages, including even programming languages in computer science.

Formulas As we have seen, predicate logic is an extension of propositional logic with *structured basic propositions* and *quantification*.

- An atomic formula consists of an n-ary predicate followed by n variables.
- A universally quantified formula consists of the symbol $\forall$ followed by a variable followed by a formula.
- An existentially quantified formula consists of the symbol $\exists$ followed by a variable followed by a formula.

- The rules for Boolean connectives are as in propositional logic.

Formal grammar We will now give a formal definition of the language of predicate logic. The definition assumes that individual terms are either variables or constants. We use a popular notation from computer science ("BNF format") for optimal perspicuity:

$$
\begin{array}{rcl}
\mathbf{v} & ::= & x \mid y \mid z \mid \cdots \\
\mathbf{c} & ::= & a \mid b \mid c \mid \cdots \\
\mathbf{t} & ::= & \mathbf{v} \mid \mathbf{c} \\
\mathbf{P} & ::= & P \mid Q \mid R \mid \cdots \\
\mathbf{Atom} & ::= & \mathbf{P}\,\mathbf{t}_1 \cdots \mathbf{t}_n \text{ where } n \text{ is the arity of } \mathbf{P} \\
\varphi & ::= & \mathbf{Atom} \mid \neg\,\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi) \mid \\
 & & \forall\,\mathbf{v}\,\varphi \mid \exists\,\mathbf{v}\,\varphi.
\end{array}
$$

Here is how you read such a schema. The lines separating items stand for a disjunction of cases. Then, e.g., the clause for formulas says that a formula is either an atomic formula, or a negation of something that is already a formula, and so on. The class of formulas is understood to be the *smallest set of symbol sequences that is generated in this way*.

The following strings are formulas of the predicate logical language:

- $\neg Px$
- $(Px \wedge Qy)$
- $((Px \wedge Qy) \vee Rxz)$
- $\forall x Rxx$
- $\exists x(Rxy \wedge Sxyx)$

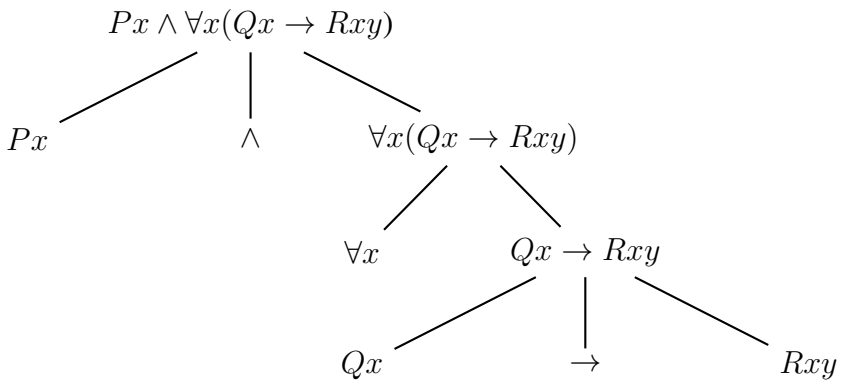
The semantics for this formal syntax will be given in the next section.

This official language definition places more parentheses than we have used so far. One usually omits parentheses when they are not essential for disambiguation.

We will now do a bit of "basic grammar" of predicate logic. The following may seem like a lot, but it is really very mild. Compare how much basic grammar you have to learn to use a natural language!

Free and bound variables Quantifiers and variables work closely together. In formulas we distinguish between the variable occurrences that are *bound* by a quantifier occurrence in that formula and the variable occurrences that are not. Binding is a syntactic notion, and it can simply be stated as follows. In a formula $\forall x\,\varphi$ (or $\exists x\,\varphi$), the quantifier occurrence binds all occurrences of x in φ that are not bound by any quantifier occurrence $\forall x$ or $\exists x$ inside φ.

As an example, consider the formula $Px \wedge \forall x(Qx \to Rxy)$. Here is its syntax tree:



The occurrence of x in Px is free, as it is not in the scope of a quantifier; the other occurrences of x (the one in Qx and in Rxy) are bound, as they are in the scope of $\forall x$. An occurrence of x is bound in φ if some quantifier occurrence binds it, and free otherwise.

Exercise 4.27 Give the bound occurrences of x in the following formula.

$$\exists x(Rxy \vee Sxyz) \wedge Px$$

Exercise 4.28 Which quantifier occurrence binds which variable occurrences?

$$\forall x(Px \to \exists x Rxx).$$

A predicate logical formula is called *open* if it contains at least one variable occurrence which is free (not bound by any quantifier); it is called *closed* otherwise. A closed predicate logical formula is also called a predicate logical *sentence*, which makes a complete assertion. $Px \wedge \exists x Rxx$ is an open formula, but $\exists x(Px \wedge \exists x Rxx)$ is a sentence.

Exercise 4.29 Which of the following formulas are sentences?

(1) $\forall x Px \vee \forall x Qx$,

(2) $Px \vee \forall x Qx$,

(3) $Px \vee Qx$,

(4) $\forall x(Px \to Rxy)$,

(5) $\forall x(Px \to \exists y Rxy)$.

If a formula φ has no free occurrences of x, quantifiers $\forall x\varphi$, $\exists x\varphi$ in it are called *vacuous*. While this may seem perverse in communication, technically, vacuous quantifiers do help in keeping the syntax and proof system of predicate logic smooth and simple.

Exercise 4.30 Which quantifications are vacuous? Replace each formula with vacuous quantification by an equivalent formula without vacuous quantification.

(1) $\forall x \exists x Rxx$.

(2) $\forall x \exists y Rxx$.

(3) $\forall x \exists y Rxy$.

(4) $\forall x \exists y Ryy$.

Substitution Now consider the ways we have for talking about objects. A *term* is either a constant or a variable. Thus, x is a term, and the constant c is also a term.

Here is a function that *substitutes* a term t for the occurrences of a variable v in a term s, with notation s_t^v:

$$\begin{array}{rcl} c_t^v & := & c \\ v_t^v & := & t \\ {v_1}_t^v & := & v_1 \text{ for } v_1 \text{ different from } v \end{array}$$

This follows the clauses constructing terms in the syntax of predicate logic.

Here is how this definition works out in a concrete case:

x_z^y is equal to x, x_c^x is equal to c, x_y^x is equal to y.

Next, we use the definition of s_t^v to define the widely used notion of *substitution of a term t for all free occurrences of a variable v in a formula φ*, with notation

$$\varphi_t^v.$$

This says that the property expressed by φ holds of the object denoted by t. This time, the definition follows the above syntactic clauses for constructing of the formulas of predicate logic:

$$
\begin{array}{rcl}
(Pt_1 \cdots t_n)_t^v & := & Pt_1{}_t^v \cdots t_n{}_t^v \\
(\neg\varphi)_t^v & := & (\neg\varphi_t^v) \\
(\varphi_1 \wedge \varphi_2)_t^v & := & (\varphi_1{}_t^v \wedge \varphi_2{}_t^v) \\
(\varphi_1 \vee \varphi_2)_t^v & := & (\varphi_1{}_t^v \vee \varphi_2{}_t^v) \\
(\varphi_1 \rightarrow \varphi_2)_t^v & := & (\varphi_1{}_t^v \rightarrow \varphi_2{}_t^v) \\
(\varphi_1 \leftrightarrow \varphi_2)_t^v & := & (\varphi_1{}_t^v \leftrightarrow \varphi_2{}_t^v) \\
(\forall v_1\varphi)_t^v & := & \begin{cases} \forall v_1\varphi & \text{if } v_1 \text{ equals } v, \\ \forall v_1\varphi_t^v & \text{otherwise.} \end{cases} \\
(\exists v_1\varphi)_t^v & := & \begin{cases} \exists v_1\varphi & \text{if } v_1 \text{ equals } v, \\ \exists v_1\varphi_t^v & \text{otherwise.} \end{cases}
\end{array}
$$

Exercise 4.31 Give the results of the following substitutions:

(1) $(Rxx)_c^x$.

(2) $(Rxx)_y^x$.

(3) $(\forall xRxx)_y^x$.

(4) $(\forall yRxx)_y^x$.

(5) $(\exists yRxy)_z^x$.

Defining notions by recursion A remarkable feature of our grammatical definitions is their use of *recursion*. We explain what $(\neg\varphi)_t^v$ means by assuming that we already know what substitution does on smaller parts: φ_t^v, and so on. This recursion works because the definition follows precisely the construction pattern that was used for defining the formulas in the first place.

Alphabetic variants Using the notion of a substitution, we can say what it means that a formula is an *alphabetic variant* of another formula. This is useful since we often want to switch bound variables while retaining the essential structure of a formula.

Suppose φ does not have occurrences of z, and consider φ_z^x, the result of replacing all free occurrences of x in φ by z. Note that $\forall z\varphi_z^x$ quantifies over variable z in all places where $\forall x\varphi$ quantifies over x. We say that $\forall x\varphi$ and $\forall z\varphi_z^x$ are *alphabetic variants*.

Here are some examples: $\forall xRxx$ and $\forall yRyy$ are alphabetic variants, and so are $\forall x\exists yRxy$ and $\forall z\exists xRzx$. The quantification patterns are the same, although different variable bindings are employed to express them.

This section has given you some basic parts of the grammar of predicate logic. There are some other grammatical notions that are widely used, such as the 'quantifier depth' of a formula, being the longest 'nesting' of quantifiers that take scope over each other inside the formula. Again, this can be defined by recursion on the construction of the formulas, as given by the grammar definition: atomic formulas have quantifier depth 0, negations do not change depth, the depth of a conjunction is calculated as the maximum of the depths of its conjuncts, and similarly for the other binary boolean connectives, and finally a quantifier increases the depth by one. The following definition expresses this formally:

$$
\begin{aligned}
d(Pt_1 \cdots t_n) &:= 0 \\
d(\neg\varphi) &:= d(\varphi) \\
d(\varphi_1 \wedge \varphi_2) &:= \max(d(\varphi_1), d(\varphi_2)) \\
d(\varphi_1 \vee \varphi_2) &:= \max(d(\varphi_1), d(\varphi_2)) \\
d(\varphi_1 \rightarrow \varphi_2) &:= \max(d(\varphi_1), d(\varphi_2)) \\
d(\varphi_1 \leftrightarrow \varphi_2) &:= \max(d(\varphi_1), d(\varphi_2)) \\
d(\forall v\varphi) &:= 1 + d(\varphi) \\
d(\exists v\varphi) &:= 1 + d(\varphi)
\end{aligned}
$$

For the moment, you have seen enough grammatical precision, and we can proceed.

4.6 Semantics of Predicate Logic

Now it is time to say in more detail how the formal language of predicate logic gets interpreted systematically on semantic structures. We have said a lot already about these situations in Section 4.4, now the time has come to be more precise. The following notion packages together the notion of "structure" that we had before with the notion of an "interpretation" linking syntactic expressions in the language to matching parts of the structure.

Models We will give our definition for a special case, but once you grasp that, you will easily see how things work in general for the formal language that we have defined in the preceding section. Let us suppose that the predicate letter P has arity 1, R has arity 2, and S has arity 3. What should a relevant situation look like? A *model* $\mathcal{M}$ is a pair (D, I) with a *domain* D consisting of individual objects, together with an I that assigns the right kind of predicates on objects in D: a property for P, a binary relation for R, and a ternary

relation for S. In set-theoretic notation, this may be written as follows:

$$\begin{aligned} I(P) &\subseteq D, \\ I(R) &\subseteq D \times D, \\ I(S) &\subseteq D \times D \times D. \end{aligned}$$

Here $D \times D$ (sometimes also written as D^2) is the set of all ordered pairs of objects in D, and $D \times D \times D$ (also written as D^3) is the set of all ordered triples of objects in D. Here is a useful notation. We write $I(P)$ as P_I, $I(R)$ as R_I, and $I(S)$ as S_I. This gives a clear distinction between a predicate letter P and the predicate P_I that interprets this predicate letter. We will not give new illustrations for such models: everything we have done in Section 4.4 is an illustration.

In general, for any predicate-logical language (with any sort of vocabulary), a model $\mathcal{M} = (D, I)$ has a non-empty domain D of objects plus an interpretation function I mapping the relation symbols of L to appropriate predicates on D, and the constants of L to objects in D.

Variable assignments Having dealt with predicates and constants, it remains to make sense of our final important piece of syntax: *variables* for objects. Intuitively, a model $\mathcal{M} = (D, I)$ suffices for giving truth values to sentences of our language, that make an assertion that is true or false in the model. But in doing so recursively, we cannot avoid dealing also with formulas containing free variables: unpeeling one quantifier will already leave us with a component formula of the latter sort.

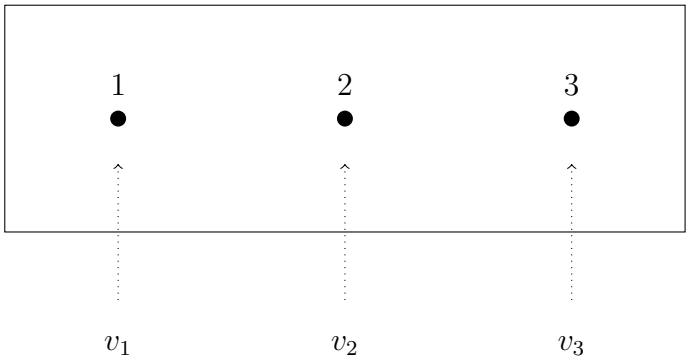
This calls for a new notion. Let V be the set of variables of the language:

> A function g from variables to objects in D is called a *variable assignment*.

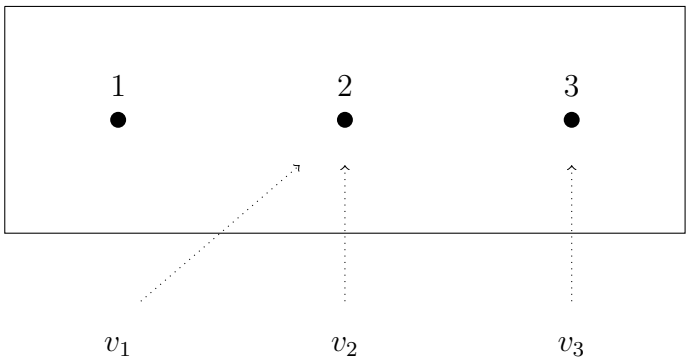
Later on, we will need to make minimal changes to variable assignments:

> $g[v := d]$ is the variable assignment that is completely like g except that v now gets the value d (where g might have assigned a different value).

For example, let $D = \{1, 2, 3\}$, and let $V = \{v_1, v_2, v_3\}$. Let $g(v_1) = 1, g(v_2) = 2, g(v_3) = 3$. See the following picture, where the g links are indicated by dotted arrows.



Then $g[v_1 := 2]$ is the variable assignment that is like g except that v_1 gets the value 2, i.e. the assignment that assigns 2 to v_1, 2 to v_2, and 3 to v_3:



Assignments and computation We emphasize this picture of value change because it has found important other applications, after it was first proposed. In particular, the above pictures drive the semantics of imperative programming languages (a topic that will be explained in Chapter 6 below). An assignment models a *memory state* of a computer, where the variables are "addresses" that store "current values". A computer typically works by successively replacing values in addresses by new values – and that is precisely what the above assignment change does.

But right here, we put assignments to work in the semantics of predicate logic.

Truth definition for predicate logic At last, we are ready for the core of the semantics of predicate logic: the definition of truth of a formula in a model. This truth definition for predicate logic is due to the Polish logician Alfred Tarski (1901–1983):

Alfred Tarski

Let $\mathcal{M} = (D, I)$ be a model for predicate logical language L, and let g be a variable assignment for L in M. Consider any formula φ of L. We will assume for concreteness that L has a unary predicate symbol P, a binary R, and a ternary S, plus a constant symbol c. The general treatment will then be obvious from what follows. We are going to define the notion

$M \models_g \varphi$, for φ *is true in* M *under assignment* g.

We also sometimes say that g *satisfies* φ *in model* M.

Values of terms First, we consider the terms of the language: variables are interpreted by the variable assignment, constants using the "hard-wired" interpretation function of the model:

$$\begin{aligned} [\![v]\!]^g_I &= g(v) \\ [\![c]\!]^g_I &= c_I \end{aligned}$$

Truth in a model Next, we define truth of a formula in a model given a variable assignment g. As usual, the definition follows the construction in the definition of predicate logical formulas in a stepwise manner:

$M \models_g Pt_1 \cdots t_n$	iff	$([\![t_1]\!]^g_I, \ldots, [\![t_n]\!]^g_I) \in P_I$
$M \models_g \neg\varphi$	iff	it is not the case that $M \models_g \varphi$.
$M \models_g \varphi_1 \wedge \varphi_2$	iff	$M \models_g \varphi_1$ and $M \models_g \varphi_2$
$M \models_g \varphi_1 \vee \varphi_2$	iff	$M \models_g \varphi_1$ or $M \models_g \varphi_2$
$M \models_g \varphi_1 \rightarrow \varphi_2$	iff	$M \models_g \varphi_1$ implies $M \models_g \varphi_2$
$M \models_g \varphi_1 \leftrightarrow \varphi_2$	iff	$M \models_g \varphi_1$ if and only if $M \models_g \varphi_2$
$M \models_g \forall v\varphi$	iff	for all $d \in D$ it holds that $M \models_{g[v:=d]} \varphi$
$M \models_g \exists v\varphi$	iff	for at least one $d \in D$ it holds that $M \models_{g[v:=d]} \varphi$

What we have presented just now is a recursive definition of truth for predicate logical formulas in given models with a given assignment. Note in particular how the clauses for the quantifiers involve the above-defined changing assignments.

Compositionality once more This formal definition implements the *compositionality* that we have discussed earlier. It shows how a small finite set of rules can interpret infinitely many formulas, of arbitrary syntactic complexity. In particular, if you think back of our earlier many-quantifier formulas, say on graphs, their meaning is completely described by the mechanism given here: *meaning of single words, repeated in tandem with formula structure*.

This completes our definition of the semantics: we now explore it a bit.

Finiteness property and closed formulas If we inspect how the truth definition works for complex formulas, the following *Finiteness Property* is easy to see:

> The truth value of a formula π in a model $\mathcal{M}$ under an assignment g only depends on the objects which g assigns to the *free variables* in φ: if two assignments agree on those free variables, they both make φ true, or both false.

In particular, then, if we evaluate closed formulas φ that have no free variables at all, they are either true under all assignments, or false under all of them. Hence we can just talk about their truth or falsity "simpliciter": writing $M \models \varphi$ iff there is *some* assignment g with $M \models_g \varphi$.

The importance of open formulas Still, even in evaluating a closed formula like

$$\forall x(Px \rightarrow \exists y Rxy),$$

the above recursive clause for the universal quantifier $\forall x$ relies on truth for the formula $Px \to \exists y Rxy$, which is open, and hence assignments are crucial to dealing with that. But more than this: formulas with free variables are not a nuisance, they are highly important in their own right. We can think of them as defining *properties of* or relations between the objects that fill the free variable slots. This is why, in computer science, they are often used to *query* a given model, say, a data base with information about individuals. In such a setting, the question $\varphi(x)?$ ("Which objects are φ-ing?") asks for a list of all objects in the model that have the property φ.

4.7 Valid Laws and Valid Consequence

Now we can look more formally at valid reasoning in predicate logic. Our earlier notions from Chapters 2, 3 return in this setting:

Valid laws A predicate logical formula φ is called *logically valid* if φ is true in every model under every assignment. Notation: $\models \varphi$.

Without further ado, here is how you can test your understanding of this:

Exercise 4.32 Which of the following statements are true?

(1) $\models \exists x Px \vee \neg \exists x Px$.

(2) $\models \exists x Px \vee \forall x \neg Px$.

(3) $\models \forall x Px \vee \forall x \neg Px$.

(4) $\models \forall x Px \vee \exists x \neg Px$

(5) $\models \exists x \exists y Rxy \to \exists x Rxx$.

(6) $\models \forall x \forall y Rxy \to \forall x Rxx$.

(7) $\models \exists x \exists y Rxy \to \exists x \exists y Ryx$

(8) $\models \forall x Rxx \vee \exists x \exists y \neg Rxy$.

(9) $\models \forall x Rxx \to \forall x \exists y Rxy$

(10) $\models \exists x Rxx \to \forall x \exists y Rxy$

(11) $\models \forall x \forall y \forall z ((Rxy \wedge Ryz) \to Rxz)$.

Incidentally, our convention that the domains of our models are always non-empty is reflected in one particular law:

$$\models \forall x \varphi \to \exists x \varphi$$

Finally, as in propositional logic, we call two formulas φ, ψ *logically equivalent* if the formula $\varphi \leftrightarrow \psi$ is valid. Many practical uses of logic consist in reducing formulas to logical equivalents that are easier to grasp or manipulate.

Valid consequence More generally, we want to study the process of valid reasoning in predicate logic. When can we draw a conclusion ψ from premises $\varphi_1, \ldots, \varphi_n$? As in earlier chapters, our answer is this. Valid consequence says that the premises can never be true while the conclusion is false, or in other words, when the truth of the premises always brings the truth of the conclusion in its wake:

> A formula ψ *logically follows from* a formula φ (alternatively, φ *logically implies* ψ) if every model plus assignment which makes φ true also makes ψ true. The notation for 'φ logically implies ψ' is $\varphi \models \psi$.

The art of testing validity How can we see that statements of the form $\varphi \models \psi$ hold or do not hold? (where φ, ψ are closed formulas of predicate logic)? Sometimes a simple informal argument confirms the validity. And in principle, it is also clear how we can *refute* the statement $\varphi \models \psi$. We have to find a *counterexample*: that is a model $\mathcal{M}$ plus assignment g with $M \models_g \varphi$, but not $M \models_g \psi$ (the latter is often abbreviated as $M \not\models_g \psi$). Using your intuitive skills of this sort, try to answer the following questions:

Exercise 4.33 Which of the following statements hold? If a statement holds, then you should explain why. If it does not, then you should give a counterexample.

(1) $\forall x Px \models \exists x Px$

(2) $\exists x Px \models \forall x Px$.

(3) $\forall x Rxx \models \forall x \exists y Rxy$.

(4) $\forall x \forall y Rxy \models \forall x Rxx$.

(5) $\exists x \exists y Rxy \models \exists x Rxx$

(6) $\forall x \exists y Rxy \models \forall x Rxx$.

(7) $\exists y \forall x Rxy \models \forall x \exists y Rxy$

(8) $\forall x \exists y Rxy \models \exists y \forall x Rxy$.

(9) $\exists x \exists y Rxy \models \exists x \exists y Ryx$.

(10) $\forall x Rxx \models \forall x \exists y Rxy$.

(11) $\exists x Rxx \models \exists x \exists y Rxy$.

We can make all this slightly more general by allowing more than one premise. We say that a formula ψ logically follows from $\varphi_1, \ldots, \varphi_n$ (in notation: $\varphi_1, \ldots, \varphi_n \models \psi$) if for every model $\mathcal{M}$ for the language and assignment g with the property that $\mathcal{M} \models_g \varphi_1$ and ... and $\mathcal{M} \models_g \varphi_n$ it is also the case that $\mathcal{M} \models_g \psi$.

Exercise 4.34 Which of the following hold?

(1) $\forall x \forall y(Rxy \to Ryx), Rab \models Rba$

(2) $\forall x \forall y(Rxy \to Ryx), Rab \models Raa$

(3) $\forall x \forall y \forall z((Rxy \wedge Ryz) \to Rxz), Rab, Rac \models Rbc$,

(4) $\forall x \forall y \forall z((Rxy \wedge Ryz) \to Rxz), Rab, Rab \models Rac$,

(5) $\forall x \forall y \forall z((Rxy \wedge Ryz) \to Rxz), Rab, \neg Rac \models \neg Rbc$,

(6) $\forall x \forall y \forall z((Rxy \wedge Ryz) \to Rxz), \forall x \forall y(Rxy \to Ryx), Rab \models Raa$.

Maybe these exercises were not too difficult, but in general, testing for predicate-logical validity is much harder than the truth table method for propositional logic in Chapter 2, or the Venn diagrams for syllogistic reasoning in Chapter 3. We will devote all of Chapter 8 later in this course to a general method for testing validity for predicate logic called *semantic tableaux*. For now, we just explain what has become more difficult than before.

Mathematical complexity of the system Here are a few reasons why predicate-logical validity is much harder than that for propositional logic. One is that there are infinitely many models for the language. This infinity arises because domains of models can have many sizes: finite $0, 1, 2, 3, ...$ (and there are already infinitely many finite natural numbers), or even infinite (the real numbers are an infinite structure, and so are the usual geometrical spaces). This means that there is no finite enumeration of all relevant models to be checked, the way we did in truth tables or Venn diagrams.

In fact, it can even be *proved mathematically* that the notion of validity for predicate logic is harder than anything we have seen before:

> Validity for predicate logic is *undecidable*: there is no mechanical method at all that tests it automatically.

A proof of this result is beyond the scope of this course, but we will make some further comments on this topic in one of the Outlooks to this chapter.

This computational complexity may be seen as the price that we pay for the nice fact that predicate logic has much greater expressive power in its language than all the logical systems we have seen before. Still, some good news remains: we *can* axiomatize the validities of predicate logic in a formal *proof system* of the sort we have seen in earlier chapters, and our next section will say a little bit about how that works. But before we go there, here is one more relevant feature to understand.

Decidable parts of predicate logic The undecidability of predicate logic has to do with its full arsenal of expressive power: especially its use of binary and higher predicates, and its complex patterns of nested quantification. What that leaves open is the possibility that, inside the complex total system, there may be large decidable parts with lower expressive power that do have decision procedures for validity, and that might still be useful. And indeed there are many such "decidable fragments" of predicate logic, with new ones being discovered all the time.

One classical example of a decidable part of predicate logic is *monadic predicate logic* (see page 4-13 above). This is the system that arises from the above presentation when we keep the syntax and semantics as we gave them, but make one sweeping restriction: all predicate letters occurring in atomic statements must be *unary*, with just one argument place. What can we say in such a fragment? Well, it is easy to see that our earlier predicate-logical translation of syllogistic forms ended up inside this fragment. But so do many other inferences about properties and kinds. Now here is a result that we state without proof:

> Monadic predicate logic has a decision method for validity.

This is actually not so hard to see, since the structure of monadic formulas is easy to analyze. In particular, one can prove that in this fragment, each formula is logical equivalent to one without nested quantifiers. As a result we can even restrict ourselves to using finite models up to a bounded size in testing validity for the monadic fragment – something you may view as a sort of extended Venn diagram method. With these perhaps too mysterious hints, we leave this topic here.

4.8 Proof

The valid laws of predicate logic can be described as the provable theorems of a formal calculus like the ones you have seen in earlier chapters. Here is its list of principles:

(1) All propositional tautologies.

(2) $\forall x\varphi \rightarrow \varphi^x_t$. Condition: no variable in t occurs bound in φ.

(3) $\forall x(\varphi \rightarrow \psi) \rightarrow (\forall x\varphi \rightarrow \forall x\psi)$.

(4) $\varphi \rightarrow \forall x\varphi$. Condition: x does not occur free in φ.

Next, the system includes a *definition* of the existential quantifier as follows:

$$\exists x\varphi \leftrightarrow \neg\forall x\neg\varphi.$$

Finally, as our deductive proof rules, we have Modus Ponens as in propositional logic, plus a rule UG of *Universal Generalization*:

- MP: from φ and $\varphi \to \psi$, infer ψ.
- UG: from φ infer $\forall x\varphi$. Condition: x does not occur free in any premise which has been used in the proof of φ.

A *theorem* is now any formula that can be derived from the given axioms and definition by the stated rules in a finite number of steps. Here are two examples:

From our Axiom 2 of "universal instantiation" we can derive a dual axiom of "existential generalization":

1.	$\forall x\neg\varphi \to \neg\varphi^x_t$	axiom 2
2.	$(\forall x\neg\varphi \to \neg\varphi^x_t) \to (\varphi^x_t \to \neg\forall x\neg\varphi)$	propositional tautology
3.	$\varphi^x_t \to \neg\forall x\neg\varphi$	from 1,2 by MP
4.	$\varphi^x_t \to \exists x\varphi$	from 3, by def of $\exists$

Our second example of using this proof system employs hypothetical reasoning to prove an implication: first derive a conclusion C from a hypothesis H, and then conclude that $H \to C$ is a theorem. We start with two hypotheses, $\varphi \to \forall x\psi$ and φ, and we assume that t x is not free in φ. Next, we drop the hypotheses one by one.

1.	$\varphi \to \forall x\psi$	hypothesis
2.	φ	hypothesis
3.	$\forall x\psi$	MP from 1,2
4.	$\forall x\psi \to \psi$	axiom 2, with x for t
5.	ψ	MP from 3,4
6.	$\varphi \to \psi$	drop hypothesis 2
7.	$\forall x(\varphi \to \psi)$	UG of 6
8.	$(\varphi \to \forall x\psi) \to \forall x(\varphi \to \psi)$	drop hypothesis 1

Note that the use of UG in step 7 is justified by our assumption that x is not free in φ, and therefore, x is not free in the hypothesis $\varphi \to \forall x\psi$. The derivation shows that

$$\vdash (\varphi \to \forall x\psi) \to \forall x(\varphi \to \psi),$$

on condition that x is not free in φ.

As earlier in this course, we are not going to train you in finding formal proofs, but you may find it rewarding to learn how to read them – and perhaps even look for some simple ones by yourself. For now, we conclude with some background.

System properties This proof system has two features that you have seen earlier. First, we have

Soundness: Every theorem of predicate logic is valid.

This is easy to see: the stated axioms are all valid, and you can check in the above list. Moreover, the proof rules are such that they derive valid formulas from valid formulas. Much more complex, and in fact about the first really deep result in modern logic is

Completeness: Every valid formula is a provable theorem.

A proof of this important result (due to Kurt Gödel in his 1930 dissertation) would typically be your passport to a next-level more advanced logic course.

Kurt Gödel

But let us end this theoretical passage with a connection with the preceding section. Predicate logic is undecidable, but it does have a complete proof system. Is not there a tension between these two results? Cannot we use the above simple proof system, at least in principle, to test for validity? Say, we write down a formula, and we *enumerate all possible proofs* in the above deductive system in a long sequence, say in ascending length. (By the way, note that this enumeration sequence will be *infinite*: there are infinitely many theorems produced by the above calculus. Why?) Now if a formula is valid, we *must* encounter it somewhere at some finite stage along this sequence, and we are done... Here is the flaw in this reasoning. A method for testing should tell us after a finite number of steps whether or not the formula is valid. But in our proof system, the only way we can see that a formula φ is *not valid* is by running through the entire infinite sequence, and noting that it nowhere lists a proof for φ. This takes infinitely many time steps, and hence it is not a decision method.

Exercise 4.35 Here is a result known as "Post's theorem": Suppose that you have an infinite enumeration of all valid formulas of a logic, and also an infinite enumeration of all non-valid formulas. Then the system has a decision method. Show how that method would work.

4.9 Identity, Function Symbols, Algebraic Reasoning

Identity First we add identity (or: 'equality') to predicate logic. We start with a simple example. Define a 'celebrity' as a person who knows no one, but is known by everyone. This means, of course, someone who does not know anyone else, but is known by everyone else. To express this we need a predicate for identity (or: equality). So let's use the predicate $x = y$ that was mentioned before. Now we can say (employing the useful abbreviation $x \neq y$ for $\neg x = y$):

x is a celebrity (i) $\forall y(x \neq y \rightarrow (Kyx \wedge \neg Kxy))$.
(ii) $\forall y(x \neq y \rightarrow Kyx) \wedge \forall y(x \neq y \rightarrow \neg Kxy)$.

The two translations (i) and (ii) express the same thing: they are logically equivalent. The following formula expresses that C is a definition of the property of being a celebrity:

$$\forall x(Cx \leftrightarrow \forall y(x \neq y \rightarrow (Kyx \wedge \neg Kxy))). \tag{4.17}$$

From definition (4.17) and the formula that expresses that Mary knows John, Kmj, it follows that $\neg Cm$ (Mary is not a celebrity). Identiy is also needed to translate the following example:

John loves Mary and Bill loves another girl. (4.18)

Clearly, with "another girl" is meant "a girl different from Mary." Using identity we can translate the example as follows:

$$Ljm \wedge \exists x(Gx \wedge x \neq m \wedge Lbx). \tag{4.19}$$

Identity is a binary predicate with a fixed logical meaning. It is always written in infix notation, as $x = y$, and with $\neg x = y$ abbreviated as $x \neq y$. Here is a similar case where identity is useful:

Mary is the only girl that John loves. (4.20)

Being the only girl with a certain property means that all other girls lack that property. Spelling this out we get:

$$Gm \wedge Ljm \wedge \forall x\,((x \neq m \wedge Gx) \rightarrow \neg Ljx) \tag{4.21}$$

By using an equivalence instead of an implication symbol we can express this a bit shorter:

$$\forall x\,(Gx \rightarrow (x = m \leftrightarrow Ljx)) \wedge Gm \tag{4.22}$$

Rephrasing this in English it says that all girls have the property that being loved by John boils down to the same thing as being equal to Mary.

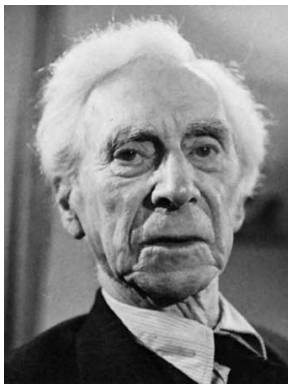
Identity provides a way of expressing *uniqueness properties* of objects. The following formula states that there is *exactly one* object which has the property P:

$$\exists x\,(Px \wedge \forall y\,(Py \rightarrow y = x)) \tag{4.23}$$

In words, there exists a 'P' such that each second object which has the property P as well must be equal to the first. Again, this can be formulated somewhat shorter by using an equivalence:

$$\exists x \forall y \, (Py \leftrightarrow y = x) \tag{4.24}$$

There is an object x such that for each object having the property P is the same thing as being equal to x.



Bertrand Russell

The British philosopher Bertrand Russell proposed to use this recipe to translate definite descriptions.

The father of Charles II was executed. (4.25)

This can be translated, using F for the property of being father of Charles II, as:

$$\exists x(Fx \wedge \forall y(Fy \rightarrow x = y) \wedge Ex), \tag{4.26}$$

or, as we have seen, more succinctly as:

$$\exists x(\forall y(Fy \leftrightarrow x = y) \wedge Ex). \tag{4.27}$$

Russell's analysis then gets extended to an account of the meaning of proper names as disguised or abbreviated descriptions.

In a similar way we can give a formula stating that there exists exactly two objects which have the property P:

$$\exists x \exists y \, (\neg x = y \wedge \forall z \, (Pz \leftrightarrow (z = y \vee z = x))) \tag{4.28}$$

This says that there exists a pair (of two *different* things) such that having the property P is the same as being equal to one of the members of this pair. To avoid such long reformulation of relatively simple information (4.28) is sometimes abbreviated as $\exists!^2 x \, Px$, and in general $\exists!^n Px$ is used to express that exactly n things have the property P. In the case $n = 1$ we also write $\exists! x \, P$.

Exercise 4.36 Write out the predicate logical translation for "there are exactly three objects with property P".

Exercise 4.37 The $\exists!^2$-quantifier can be used to give a predicate logical description of prime numbers in the divisibility graph as depicted in (4.16) on page 4-22. How? And what kind of numbers do you get when you replace $\exists!^2$ in your definition by $\exists!^3$?

Exercise 4.38

(1) Explain why the two following sentences have different meanings (by a description of a situation in which the sentences have a different truth-value)

There is exactly one boy who loves exactly one girl. ($\exists!x\,\exists!y\,(Bx \wedge Gy \wedge Lxy)$)

There is exactly one girl who is loved by exactly one boy. ($\exists!y\,\exists!x\,(Bx \wedge Gy \wedge Lxy)$)

(2) Explain why $\exists!x\,(Ax \vee Bx)$ and $\exists!x\,Ax \vee \exists!x\,Bx$ have different meanings.

Function symbols There is still one other device that predicate logic has for speaking about objects: function symbols. Mathematics is replete with *functions* that create new objects out of old, and likewise, predicate logic becomes more expressive if we allow function symbols for functions like 'the square of x' or 'the sum of x and y'. We will see later how this can be used to write algebraic facts and do calculations, but for starters, note that function symbols also make sense for natural language. On the domain of human beings, for instance, the *father* function gives the father of a person. Suppose we write $f(x)$ for the father of x and $m(x)$ for the mother of x. Then we can say things like:

Paula's father loves her	$Lf(p)p$
Bill and John are full brothers	$f(b) = f(j) \wedge m(b) = m(j)$
Not everyone has the same father	$\neg\forall x\forall y f(x) = f(y)$

In the lingo of mathematics, function symbols are combined with equality to express algebraic equations like the following:

$$x \cdot (y + z) = x \cdot y + x \cdot z. \tag{4.29}$$

This is a statement of the sort that you learn to manipulate in high school. In fact, this can be viewed as a formula of predicate logic. There is the equality predicate here, but the main point is made with the function symbols for addition and multiplication.

Finally, let us see what changes to the truth definition are needed to take the identity relation and function symbols into account. Identity statements are treated by adding the following clause to the semantics definition:

$$M \models_g t_1 = t_2 \quad \text{iff} \quad [\![t_1]\!]^g_I = [\![t_2]\!]^g_I$$

This treats identity as a special binary predicate, with a fixed interpretation: $t_1 = t_2$ is true if and only if t_1 and t_2 get interpreted as the same object in the model under consideration.

To extend the semantics to function symbols, we have to extend the truth definition to structured terms (terms containing function symbols). For that, we have to assume that the interpretation function I knows how to deal with a function symbol.

If f is a one-place function symbol, then I should map f to a unary operation on the domain D, i.e. to a function $f_I \colon D \to D$. In general it holds that if f is an n-place function symbol, then I should map f to an n-ary operation on the domain D, i.e. to a function $f_I \colon D^n \to D$. We use C for the set of zero-place function symbols; these are the constants of the language, and we have that if $c \in C$ then $c_I \in D$.

The interpretation of terms, given an assignment function g for variables and an interpretation function I for constants and function symbols, is now defined as follows. Note the by now familiar use of recursion in the definition.

$$[\![t]\!]^g_I = \begin{cases} t_I & \text{if} \quad t \in C, \\ g(t) & \text{if} \quad t \in V, \\ f_I([\![t_1]\!]^g_I, \ldots, [\![t_n]\!]^g_I) & \text{if} \quad t \text{ has the form } f(t_1, \ldots, t_n). \end{cases}$$

Algebraic reasoning In a domain of numbers (natural numbers, integers, fractions, reals) $+$ is a function symbol that takes two numbers and creates a number. The symbol $\cdot$ (also written as $\times$) is also a binary function symbol for the same domain of discourse.

Algebraic equations typically say that two complex terms, viewed as two different instructions for computation, denote the same object. If x, y, z refer to numbers, then $x \cdot (y + z)$ also refers to a number, and equation 4.29 given above states that $x \cdot y + x \cdot z$ refers to the same number.

Figure 4.1 gives a general set of equations that describes the interaction of addition and multiplication in a so-called *ring*, for a language with constants 0 and 1, and with binary functions $+$ and $\cdot$ (called binary operators) and unary function $-$. It is left to you to check that the equations hold, if we interpret the variables as integer numbers, but also if we interpret them as floating point numbers (fractional numbers, rational numbers), and also if we interpret the variables as real numbers (but do not worry if the difference between rational and real numbers escapes you).

$$
\begin{aligned}
x + 0 &= x \\
x + (-x) &= 0 \\
x + y &= y + x \\
x + (y + z) &= (x + y) + z \\
x \cdot 1 &= x \\
1 \cdot x &= x \\
x \cdot (y \cdot z) &= (x \cdot y) \cdot z \\
x \cdot (y + z) &= x \cdot y + x \cdot z \\
(x + y) \cdot z &= x \cdot z + y \cdot z
\end{aligned}
$$

Figure 4.1: Axioms for the interplay of $+$ and $\cdot$ in rings.

$$
\begin{aligned}
x \vee y &= y \vee x \\
x \wedge y &= y \wedge x \\
x \vee (y \vee z) &= (x \vee y) \vee z \\
x \wedge (y \wedge z) &= (x \wedge y) \wedge z \\
x \wedge (x \vee y) &= x \\
x \vee (x \wedge y) &= x \\
x \wedge (y \vee z) &= (x \wedge y) \vee (x \wedge z) \\
x \vee x' &= 1 \\
x \wedge x' &= 0 \\
x \vee 1 &= 1 \\
x \wedge 0 &= 0
\end{aligned}
$$

Figure 4.2: Laws of Boolean Algebra.

The same syntax with function symbols works everywhere in mathematics. Think of mathematical notation for sets. The domain of discourse now is a universe of sets. Letters x, y, ..., stand for arbitrary sets, $\emptyset$ is the special constant name of the empty set, and function terms for intersection and union create complex terms like $x \cap (y \cup z)$ that denote sets in the way you have already seen when computing pictures for Venn Diagrams and related structures.

We can also look at propositional logic in an algebraic way. The equations in Figure 4.2 describe the laws of Boolean algebra, with binary operators $\wedge, \vee$, a unary operator $'$ for complement (negation), and constants 1 and 0. It is left to you to check that these equations are indeed valid laws of propositional logic, if we read identity as propositional equivalence, 1 as truth, and 0 as falsity.

4.10 Outlook — Mathematical Background

Using the proof system we presented in Section 4.8, you can talk about all models of predicate logic, for a given choice of predicate letters and function symbols. But it is also common to *add* axioms, in order to talk about specific topics. As you may already have guessed, predicate logic can be used to talk about *anything*.

Mathematical Theories: Arithmetic Suppose we want to talk about addition on the natural numbers. This is the aritmethic of first grade, the stuff that comes before the tables of multiplication. The domain of discourse is

$$\mathbb{N} = \{0, 1, 2, 3, 4, \ldots\}.$$

The predicate logical language that we can use for this has a constant for zero (we will use $\mathbf{0}$ for this; the intention is to use $\mathbf{0}$ as a name for the number 0), and two function symbols, one for taking the successor of a number (we will use s for this), and the familiar $+$ for addition. The successor of the number is the number immediately following it.

We have four axioms and an axiom scheme (a recipe for constructing further axioms).

The first axiom states that 0 is not the successor of any number:

$$\forall x \, \neg sx = \mathbf{0}. \tag{PA1}$$

The second axiom states that different numbers cannot have the same successor, or stated otherwise, that if $sx = sy$ then $x = y$.

$$\forall x \forall y (sx = sy \to x = y). \tag{PA2}$$

Note that $sx = sy \to x = y$ is equivalent to $x \neq y \to sx \neq sy$, so this does indeed express that different numbers have different successors.

The third axiom expresses that adding zero to a number doesn't change anything:

$$\forall x\ x + \mathbf{0} = x. \tag{PA3}$$

The fourth axiom expresses a sum of the form $x + sy$ as the successor of $x + y$:

$$\forall x \forall y\ x + sy = s(x + y). \tag{PA4}$$

The third and fourth axiom together define addition for any pair of numbers x and z, as follows. Either z is equal to 0, and then apply (PA3) to see that the outcome is x, or z is the successor of another number y and then apply (PA4) to see that the outcome is the successor of $x + y$. This is called a *recursive* definition, with recursion on the structure of y.

Below we will see how the recursion definition of addition will help us in constructing inductive proofs of facts about the addition operation.

The final axiom takes the form of a scheme. Assume that $\varphi(x)$ is a formula with x free. Then the following is an axiom:

$$(\varphi(\mathbf{0}) \wedge \forall x(\varphi(x) \to \varphi(sx))) \to \forall y \varphi(y). \tag{PA5–scheme}$$

This axiom scheme expresses mathematical induction on the natural numbers. If you are unfamiliar with mathematical induction you might wish to consult section A.6 in the Appendix.

The theory of arithmetic defined by (PA1) — (PA5-scheme) is known as *Presburger arithmetic*, after Mojzesz Presburger, student of Alfred Tarki.

Mojzesz Presburger

In the language of Presburger arithmetic one cannot express properties like being a prime number, or define the relation of divisibility. But one can express properties like being

even, being odd, being a threefold, and so on. As an example, the following formula of Presburger arithmetic expresses the property of being even:

$$\exists y\ y + y = x. \qquad (x \text{ is even})$$

And the property of being odd:

$$\exists y\ s(y + y) = x. \qquad (x \text{ is odd})$$

Exercise 4.39 Express the property of x of being a threefold in the language of Presburger arithmetic.

Adding multiplication Presburger arithmetic is the restriction of the arithmetical theory of addition and multiplication to just addition. If we extend the language with an operator $\cdot$ for multiplication, the properties of multiplication can be captured by a recursive definition, as follows:

$$\forall x\ x \cdot \mathbf{0} = \mathbf{0}. \qquad \text{(MA1)}$$

$$\forall x \forall y\ x \cdot sy = (x \cdot y) + x. \qquad \text{(MA2)}$$

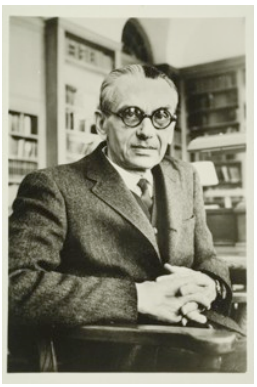
This defines $x \cdot y$ by recursion on the structure of y: MA1 gives the base case, MA2 the recursion case. In still other words: MA1 gives the first row in the multiplication table for x, and MA2 gives the recipe for computing the next row from wherever you are in the table, as follows:

$$\begin{array}{l} 0 \\ x \\ x + x \\ x + x + x \\ \vdots \end{array}$$

As you may have realized in second grade, the multiplication tables are constructed by means of addition.

Next, let IND be the induction axiom scheme for the language extended with multiplication. Thus, IND looks like PA5-scheme above, but for the extended language.

Giuseppe Peano



Kurt Gödel

The theory consisting of PA1 – PA4 plus MA1, MA2 and IND is called *Peano arithmetic*, after Giuseppe Peano (1858–1932). Here the logical situation is dramatically different from the case of Presburger arithmetic. One of the most famous results in logic is Kurt Gödel's incompleteness proof (1931) for this predicate logical version of arithmetic:

> Any predicate logical theory T that is an extension of Peano arithmetic is incomplete: it is possible to find formulas in T that make true statements about addition and multiplication on the natural numbers, but that cannot be proved in T.

Alonzo Church

Alan Turing

Five years later Alonzo Church and Alan Turing proved (independently) that the predicate logical language of arithmetic is undecidable, and more generally, that any predicate logical language with at least one 2-place relation symbol is undecidable (see Chapter 10).

4.11 Outlook — Computational Connection

Domains of discourse in programming The domains of discourse of predicate logic also play an important role in programming. In many programming languages, when a programming variable is declared, the programmer has to specify what kind of thing the variable is going to be used for.

In programming terms: variable declarations have to include a *type declaration*. A Java declaration `int i j max` declares three variables of type *int*, and an *int* in Java is a signed integer that can be stored in 32 bits: Java *int* variables range from $-2^{31} = -2,147,483,648$ to $2^{31} - 1 = 2,147,483,647$. What this means is that the type declarations fix the domains of discourse for the programming variables.

In programming, *types* are important because once we know the type of a variable we know how much storage space we have to reserve for it. More importantly, type information can be used to find common programming errors resulting from instructions to

perform operations that violate type constraints, such as an attempt to add a character to an integer. Typing information for strongly typed languages such as Haskell is expressed in (a fragment of) predicate logic.

Statement of pre- and postconditions Predicate logic can also be used to state pre- and postconditions of computational procedures. Consider the following function (in Ruby) for computing an integer output from an integer input, together with a precondition and a postcondition:

```
# precondition: n >= 0
def ld(n)
 d = 2
 while d**2 <= n
   return d if n.remainder(d) == 0
   d = d + 1
 end
 return n
end
# postcondition:
# ld(n) is the least positive integer that divides n
```

The precondition is a predicate logical formula without quantifiers:

$$n \geq 0.$$

The postcondition can be translated into a predicate logical formula, using $|$ for divides, as follows (we assume that the domain of discourse is $\mathbb{Z}$, the domain of integers):

$$\text{ld}(n)|n \wedge \forall m(0 < m < \text{ld}(n) \rightarrow \neg m|n).$$

The intended meaning is: if the input of the function satisfies the precondition then the output of the function will satisfy the postcondition.

Pre- and postconditions can be used for proving computational procedures correct. Examples of such correctness reasoning will be discussed in section 6.10.

Explicit statement of pre- and postconditions is also the key ingredient of a programming style called *design by contract*, the idea being that the precondition makes explicit what a computational procedure may assume about its input, while the postcondition is a statement about what the procedure is supposed to achieve, given the truth of the precondition. Preconditions state rights, postconditions list duties.

Exercise 4.40 (You should only attempt this if you have some programming experience.) Suppose the procedure for `ld` would have been stated like this:

```
# precondition: n >= 0
def ld(n)
 d = 2
 while d**2 < n
   return d if n.remainder(d) == 0
   d = d + 1
 end
 return n
end
# postcondition:
# ld(n) is the least positive integer that divides n
```

Why would this version of the program not satisfy the contract anymore? Try to find a simple counterexample.

4.12 Outlook — Predicate Logic and Philosophy

Historically, the development of predicate logic has had considerable influence on philosophy. The founding father of modern predicate logic, Gottlob Frege, was deeply convinced that ordinary language is inadequate for rigorous scientific thinking. Here is his famous comparison between ordinary language and his 'ideography' (what he means is: the formal language he proposes, which is essentially the language of predicate logic):

> I believe that I can best make the relation of my ideography to ordinary language clear if I compare it to that which the microscope has to the eye. Because of the range of its possible uses and the versatility with which it can adapt to the most diverse circumstances, the eye is far superior to the microscope. Considered as an optical instrument, to be sure, it exhibits many imperfections, which ordinarily remain unnoticed only on account of its intimitate connection with our mental life. But, as soon as scientific goals demand great sharpness of resolution, the eye proves to be insufficient. The microscope, on the other hand, is perfectly suited to precisely such goals, but that is just why it is useless for all others. [Fre76]

Frege believes that his proposal of a language of thought is important for philosophy precisely because it exhibits the limitations of ordinary language:

> If it is one of the tasks of philosophy to break the domination of the word over the human spirit by laying bare the misconceptions that through the use of language often almost unavoidably arise concerning the relations between concepts and by freeing thought from that with which only the means of expression of ordinary language, constituted as they are, saddle it, then my

> ideography, further developed for these purposes, can become a useful tool for the philosopher. [Fre76]

This point of view has been enormously influential. Philosophers like Bertrand Russell and Willard Quine concluded from the advances in logic that they had witnessed in their days that an important task for philosophy is to point out and repair fallacies that are due to the sway of natural language over the mind.

To focus on a light-hearted example, consider the following famous exchange:

> 'I see nobody on the road,' said Alice.
> 'I only wish I had such eyes,' the King remarked in a fretful tone. 'To be able to see Nobody! And at that distance too!'
> *Alice in Wonderland* [Car65]

The natural language syntax for "Nobody is on the road" sees this as a subject "Nobody" combined with a predicate "is on the road". But this natural language syntax does not correspond to the logical form, for the predicate logical translation of the sentence has no constituent corresponding to the subject:

$$\neg\exists x(\text{Person}(x) \wedge \text{OnTheRoad}(x)).$$

Though Frege's views fit in a long tradition of distinguishing between linguistic 'surface form' and its underlying 'logical form', Russell, Quine and other famous philosophers saw it as a battle call. They took this to mean that classical philosophy needed a total overhaul, since many of its assertions (say metaphysical ones about what sort of abstract objects exist in the world) were deeply infected with uncritically adopted natural language. Whether this accusation was true, is of course another matter. In fact, right until the present, there remains a difference in style between 'formal language philosophers' and 'natural language philosophers'.

Incidentally, in modern approaches to natural language analysis this problem is resolved by giving the subject a more articulate treatment: "nobody" refers to the set of all properties that no person has:

$$\{Y \mid \neg\exists x(\text{Person}(x) \wedge Y(x)\}. \tag{4.30}$$

Since "being on the road" obviously refers to a property, we can view the syntactic operation of combining a subject with a predicate as composition of a function with its argument. This function checks whether the property of being on the road is among the properties listed in (4.30). This being said, it remains the case that quantifier expressions in natural language may behave differently from what you would expect after this training in predicate logic. A well-known peculiarity is that "a" may also be *generic*, with universal force. This would be written in logic with a universal quantifier:

A dog has fleas	$\forall x(Dx \rightarrow \exists y(Fy \wedge Hxy))$
A banker is trusted by no-one	$\forall x(Bx \rightarrow \forall y\neg Tyx)$.

It is quite puzzling that such generic force may result from the context in which an indefinite appears. Here is a famous example:

If a farmer owns a donkey he beats it. $\forall x \forall y((Fx \wedge Dy \wedge Oxy) \rightarrow Bxy)$.

Moving beyond philosophical disputes, a whole discipline of formal analysis of natural language has formed to attempt to solve such puzzles and create insight in how humans use language to convey meaning.

Summary of Things You Have Learnt in This Chapter *You have learnt a rich new logical language, which you can use to analyze mathematical but also natural language, as has been shown in many examples going from sentences to logical forms. You have learnt precisely how such a language gets a recursive compositional semantics in models with objects and predicates that can often be pictured very concretely. This is useful beyond this chapter, since this method has become a paradigm for formal semantics of languages in general. You have also acquired a first sense of validity and what is valid and invalid reasoning in this language. And finally, you have seen some glimpses of why this system is computationally more complex than what you ad learnt so far, illustrating the balance between expressive power and computational complexity that is the art underlying logic today.*

Further Reading Modern logic begins with the proposal for a formal language of pure thought by the German mathematician and philosopher Gottlob Frege. See [Fre67], in [Hei67]. Or if you read German: [Fre79]. Frege does not yet make a clear distinction between first order logic and higher order logic. The German mathematician David Hilbert seems to have been the first to stress this distinction. See the classic textbook [DA28] (in German), or its English translation [DA50]. Many excellent treatments of predicate logic exist, with emphasis on connections with philosophy [Hod01], with linguistics [Gam91], with mathematics [CH07], or with computer science [Bur98, HR04]. A comic style account of the development of modern logic can be found in [DP09] (also see `www.logicomix.com`).

Knowledge, Action, Interaction

Introduction to Part II

In the first part of this course, you have learnt three basic logical systems that offer a steadily richer view of what the world is like: propositional logic, the syllogistic, and predicate logic. In presenting things this way, we have emphasized the role of logic in describing truth and falsity, and the valid consequences that can be drawn about how things are, or are not.

But a language is not just a medium for describing the world. Its primary function is as a medium of communication, and other activities where language users are involved. As we have seen in the Introduction to this course, we are then in the realm of conversation, argumentation, and similar logical activities. When we take these seriously, several new themes become important. Perhaps the most important change is this: in addition to languages, we need to talk about their *users*, so our logics must deal with what are nowadays often called "agents". It is their activities that need to be brought to the fore. And once we do that, we find three fundamental topics. The first is that in addition to truth, there is *information*, the real fuel that drives agents' reasoning and action. Next, those *actions* themselves are our second basic theme: agents act, thereby changing the world, as well as what they know or believe about it. And the third theme tied up with this is *social interaction*, the fact that most interesting forms of action and information flow involve many agents responding to each other, and achieving goals through that interaction.

One of the surprising things about modern logic is that it is actually able to study all these phenomena in the same precise manner that you have seen so far. And what makes this course different from the usual ones is that we are going to teach you how. Again, there will be three chapters, covering the three major aspects of "logic in action". Chapter 5 is about information and the knowledge that agents have on the basis of that, using a system first invented by philosophers called *epistemic logic*. Chapter 6 is about action in general, using a system first invented by computer scientists, called *dynamic logic*. And finally, Chapter **??** looks at multi-agent social interaction in the form of *games*, giving you an impression of the current contacts between logic and game theory.

Once you have grasped this second set of chapters, and in combination with the more classical part that you have learnt before, you will have a true feeling for what logic can do, and how it can be close to activities that you experience on a daily basis. In later parts of this course, we will then put in place further techniques that help you deepen all this.

Chapter 5

Logic, Information and Knowledge

Overview The first part of this course has shown how logical systems describe the world using objects, predicates, quantifiers and propositional combinations. This information about the world is typically conveyed when we use language, and such information then leads to knowledge among language users. This chapter deals with the logic of knowledge as based on information, including changes in knowledge which result from observations of facts, or communication between agents knowing different things. This area is called *epistemic logic*, and its main difference with the earlier systems of Chapters 2, 3 and 4 is that we can also express facts about knowledge of one or more agents in the logical language itself. This 'social' perspective occurs in many settings: knowing what others do or do not know determines our actions. Another central theme of this chapter is "change": successive information processing steps change what agents know, and this, too, is essential to understanding the logic of language use and other cognitive tasks.

5.1 Logic and Information Flow

From truth to informational actions by agents In this course, we have explained a valid argument such as

$$\text{from } p \to q \text{ and } \neg q \text{ to } \neg p$$

as:

> whenever $p \to q$ is true and $\neg q$ is true, $\neg p$ is also true.

But, true for whom? If we think about how a logical system is used, there is usually a knowing subject performing the inference, giving it a character more like this:

> If I know $p \to q$ and I know $\neg q$, then I also know $\neg p$.

And there need not be just me. Suppose I know that $p \to q$, while you do not know this, but you do know that $\neg q$. Then we should be able to pool our knowledge to reach the conclusion $\neg p$. What informational actions are involved here?

Recall what we explained in the introductory chapter of this book. Information can come from different events. The three main sources that have long been recognized for this are

observation, inference and *communication*.

Agents may come to know directly that propositions are true by perception, they can infer propositions from things they already know, and they can also learn things from others.

Example 5.1 Here is a story which ancient Chinese logicians used already some 2500 years ago to make this point:

> Someone is standing next to a room and sees a white object outside. Now another person tells her that there is an object inside the room of the same colour as the one outside. After all this, the first person knows that there is a white object inside the room. This is based on three actions: an observation, then an act of communication, and finally an inference putting things together.

Being explicit about what agents know The logical systems that you have learnt in this course can express the factual content of the information that we have just discussed. They can even model information flow through the steps of inference or update in propositional logic. But if we want to bring more of this information flow into a logical system, we need to talk about agents. This is even more pressing since we often reason explicitly about other agents' knowledge. We ask questions to people when we think they may know the answer, and we are also interested in what other people do not know, like when a teacher is trying to tell the students something new.

All this may sound simple and familiar, but it also poses interesting challenges. I tell you: "You *don't know* it, but these days I live in Seville." Having now told you this, you *know* that I live in Seville! So, what I told you has become false precisely because I said it. Is this not a paradox? One of the things you will see in this chapter is why this is not paradoxical, but rather a typical point in the logic of communication.

This *epistemic logic* of this chapter starts from propositional logic, but now enriched with logical operators $\Box_i \varphi$ (also written als $K_i \varphi$) standing for the natural language expression "agent i knows that φ". Once we have this system in place, we will also use it for a description of information flow by observation and communication.

The subjects whose knowledge we describe are called *agents*. These agents can be human beings, but also measurement devices, or information processes running on a computer, each with their own informational view of the total situation. A system modeling the interaction of different agents is often called a *multi-agent system* . The agents of such

a system can be a group of people having a conversation, but also the many computers making up the Internet, or in more physical terms, the players of a soccer team with their varying fields of vision and varying abilities to change their positions.

Exercise 5.2 Imagine players in a soccer match. What channels of information are available to them? What factors restrict the information flow? (If you don't like soccer, then you are invited to replace this example with your own favourite game.)

5.2 Information versus Uncertainty

Before we give our logical system, let us first ask a preliminary question:

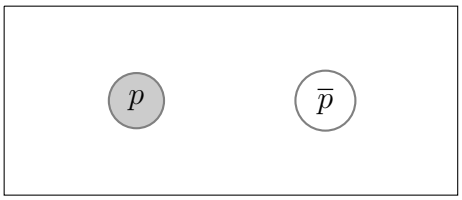
What is information?

Actually, this is not easy to say, and many disciplines in the university have things to contribute. But here is a basic idea that occurs widely: we approach things in the converse order, and model not information but *uncertainty*. And the idea for that is simple. To a first approximation, this idea can be stated with the semantic valuations for propositional logic that you have learnt. There is one actual world or actual situation, but we may not yet know what it is, and hence we must consider a larger range of options:

Uncertainty is the set of current options for the actual world.

Information models: sets of possible situations What these options are depends on the concrete scenario that we have in mind. Let us look at some examples.

Example 5.3 (Two options for one agent.) You are about to kick a ball. You may either score a goal, or not. The two options 'score', 'not-score' are the two relevant situations, which you can represent as the two valuations for an atomic statement p = 'I score a goal', one with $V(p) = 1$ and one with $V(p) = 0$. The same pattern can of course happen in many other scenarios: 'Pass' or 'Fail' for an exam that you have taken, 'Head' or 'Tails' for the outcomes of the next throw of a coin, 'Left' or 'Right' for the correct way to the Rijksmuseum, and so on. It is often easiest to think of this in the form of a *picture*. In what follows, the circles stand for the possibilities (for which we also use the term *worlds*), the proposition letters indicate which atomic facts are true where, and the shaded circle indicates the actual situation.

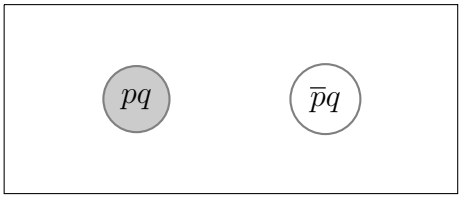


In this picture, p is true, but there is uncertainty about this: the actual situation cannot be distinguished from a situation where p is false (indicated by $\overline{p}$). Later on in this chapter, we will be more precise about the definition of such pictures, and how they can be viewed as information models.

So far, we have left something out that is often important to make explicit. We assume that there always is an *actual situation*, the reality we live in. That means that one of the possibilities in the model is the 'actual' or 'real' one. Of course, the agent herself need not know which of the various possibilities this is: if she did, she would know more than what we have pictured! One often indicates this actual world in some special way. Think of it as a marking that is invisible to the agents inside the model. It may be put there by an omniscient outside observer, or by you as the designer of the scenario. In our pictures, the actual world is usually marked by shading it grey. In the above picture, therefore, the actual world is the one to the left, and the truth of the matter is that p holds.

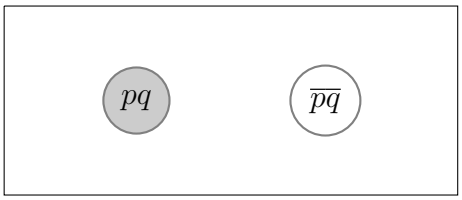
It seems that there was no information at all in the preceding model. But things soon get more interesting.

Example 5.4 (Ignorance and knowledge.) Suppose that your current range of options is the two valuations $\{V, V'\}$, with $V(p) = V(q) = 1$ and $V'(p) = 0$, $V'(q) = 1$.



Intuitively, you still know nothing about p, but you do *know that q is the case*, since it is true in every possibility that you consider.

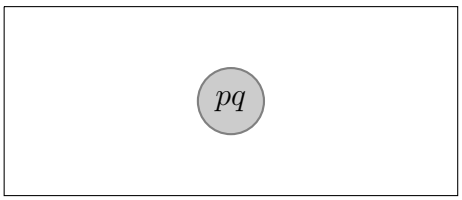
Or things could even be a bit more delicate. This time, your current range of options is again two valuations $\{V, V'\}$, but now with $V(p) = V(q) = 1$ and $V'(p) = V'(q) = 0$.



Now you do not know whether p is true, or whether q is true, but you do know that one is true if and only if the other is. In particular, if you can find out the truth value of p, say by making an observation or asking some trusted person, you will automatically know the truth value for q. This is indeed how information works: Suppose you know that Mary

and John are a dancing couple, and that either both of them show up in the ballroom or neither. If someone tells you that they saw Mary, you conclude that John was there as well.

As a very special case, a model with just one option represents a situation of complete information about what things are really like. In the following picture, the agent knows what the actual world is, and in particular, that both p and q are true there:



One set of possibilities, as we have considered so far, just describes what one particular agent knows and does not know. Now let us look at a typical situation where more agents are involved. Suppose that

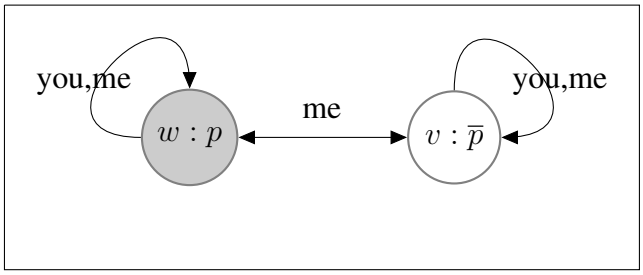
> I do not know if p is the case, but I know that you know *whether* p is the case (i.e., if p is true, you know that, and if p is false, you know that, too).

This would be a good reason, for instance, for me to ask you a *question*, namely, "Is p the case?". How can we model this social scenario?

Example 5.5 (Two options and two agents.) There are two possibilities for whether p is the case or not; let us call them p, $\overline{p}$ for short. But this time, there is a distinction between how I see them and how you see them. For me, both options are possible, so they belong to the same set $\{p, \overline{p}\}$. To indicate that I cannot distinguish between these cases, we can draw a connecting arrow marked 'me'. But for you, the two possibilities do not belong to the same set, since your knowledge actually allows you to distinguish them. That is, you distinguish the two sets $\{p\}$ and $\{\overline{p}\}$. For you, there is no connecting arrow between the two possibilities.

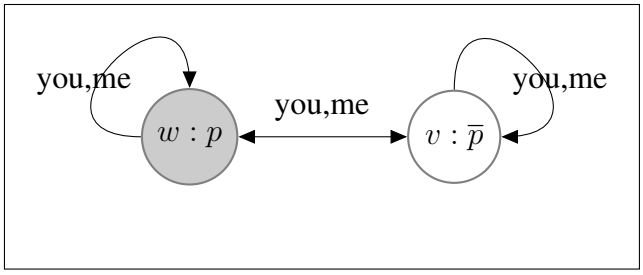
Now, a picture really helps to visualize what is going on. We draw the two relevant options as points w, v with the truth or falsity of p indicated.

But this time there is a new feature. The *line with the two arrowheads* in the picture indicates which situations agent 'me' cannot distinguish. More precisely, I see w connected to v by a line marked with the 'label' *me* because both are options for me. But you have no line there, since you are informed about the distinction between p and $\overline{p}$.



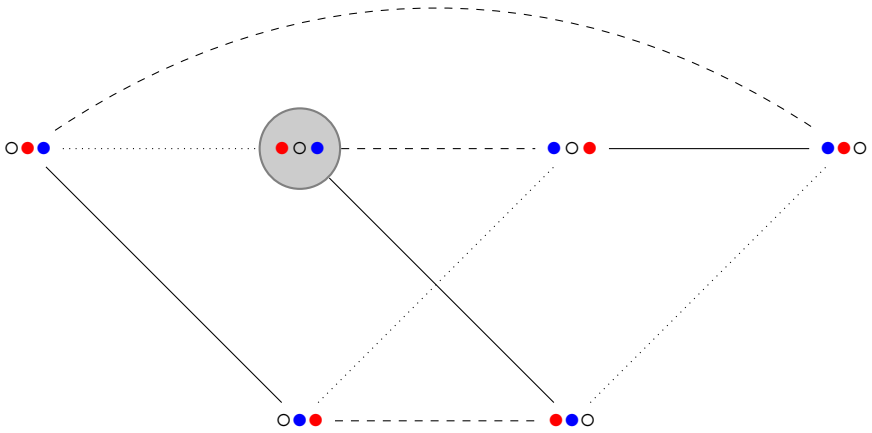
What about the loop arrows drawn for you and me? These indicate that we cannot distinguish an option from itself. That sounds obvious, but we will soon see what this means for the logic of knowledge. (In the rest of this chapter, to avoid cluttering pictures, we will often drop such loop arrows, taking them as tacitly understood.)

If we were to add a labeled arrow between the two worlds for the agent you to this picture, the model would show that neither agent knows if p is the case:



Beyond these simple examples, more interesting scenarios arise in simple stories with *cards*. Parlour games are a concrete "information lab" with agents knowing different things.

Example 5.6 Consider the start of a very simple card game. Three cards red, white, blue are given to three players: 1, 2, 3, one each. Each player sees her own card, but not that of the others. The real distribution over the players 1, 2, 3 (the "deal") is red, white, blue (●○●, or rwb). Here is a picture of the information model:



To avoid cluttering the picture, self loops and arrow points are omitted (but assumed to be there). The solid links are the links for player 1, the dashed links are the links for player 2, and the dotted links are the links for player 3. The 6 worlds are the 6 relevant options, which are the 6 possible deals of the cards (3 cards over 3 people), with the appropriate uncertainty links between deals, marked for players. For instance, the solid line between ●○● (rwb) and ●●○ (rbw) indicates that player 1 cannot distinguish these two situations, whereas 2 and 3 can: if 1 has the red card, she is uncertain about whether 2 and 3 have white and blue or blue and white respectively; however, there is no such uncertainty for 2 and 3, because 2 and 3 both know which card they have themselves.

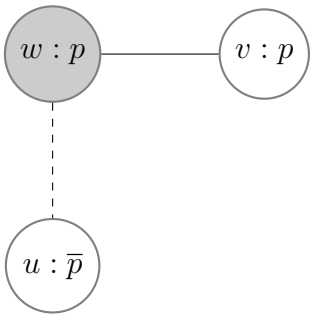
In words, the diagram says things like the following. In the actual world, agent 1 knows that she has the red card, since this is the case in the only two options that she considers possible. About the other agents, the only thing agent 1 knows is that they must have either blue or white, but she does not know in which order. As for the status of the shaded world ●○● (rwb): though the players are actually in ●○● (as an outside observer could see), none of 1, 2, 3 knows this. So, if they are to find out more, further information will have to come in. That is of course precisely the point of card games, and we will continue with this example later on.

You may already have recognized the structures that we are drawing here. They are *undirected graphs* in the sense of Chapter 4, consisting of points (the possibilities) and uncertainty lines for agents. So far, these lines have no direction: two points being indistinguishable is a symmetric relation. But in modeling other forms of information, it also makes sense to have directed graphs with arrows, as we will see later.

These examples were simple, but they should give the main idea. For now we conclude with a slightly more tricky example. We have suggested that possibilities are like propositional valuations. But this is not always completely true.

Example 5.7 (Knowledge about others.) Consider the following information model with two kinds of links for different agents. The solid links are for agent 1 and the dashed links

for agent 2.



Here the two worlds w, v have the same valuation with p true. And yet they are not identical. Why? Look at the actual world w. Agent 1 sees only the two worlds w, v there (indicated by the solid line), and hence she knows that p. But she is uncertain about what agent 2 knows. In world w, agent 2 does not know that p (he also considers the $\neg p$-world u possible, as indicated by the dashed line), but in world v, agent 2, too, knows that p (he has no uncertainty lines connecting world v to other worlds). Since w is the actual world, the truth of the matter is that 1 knows that p, and 2 does not, but 1 is not sure about what 2 knows. In particular, 1 considers it possible that 2 knows that p, but 1 also considers it possible that 2 does not know that p. How could such a situation come about? Well, for instance, 1 knows that p because she heard the teacher say that p, but she was not sure if the other student 2 was paying attention.

Many situations in real life are like this. Maybe you already know that this stove is hot, maybe you do not, but I know that it is. For safety's sake, I now tell you that it is hot, to make sure you do not hurt yourself. We will soon see how to reason about such subtle differences in information in a precise manner.

Exercise 5.8 Find a concrete practical situation where what matters to my actions is three repetitions of knowledge about other agents: what I know about what you know about my knowledge.

Information and knowledge in natural language We have given pictures for information of one or more agents. But of course we also use natural language to talk about information, both about the facts and about others. This is just the point of expressions like "John knows that Mary has taken his car", or "John does not know that Mary knows his pin code". We can introduce a convenient shorthand notation for this as follows:

$\Box_i\varphi$ for 'agent i knows that φ'.

An alternative for this is to use $K_i\varphi$ for $\Box_i\varphi$. The letter K stands for the k in the word "knowledge". In this book we will use $\Box_i$ rather than K_i. The operator $\Box_i$ comes with a dual $\Diamond_i$ (see below). (If there is just a single agent we use $\Box\varphi$ for "the agent knows that φ".)

This new operator $\Box_i$ can be freely combined with the logical languages that you have already learnt to form many types of expression involving information. For example, John (j) knows *whether* Mary has taken his car (p) if he knows which is which, so this sentence corresponds to the formula

$$\Box_j p \vee \Box_j \neg p.$$

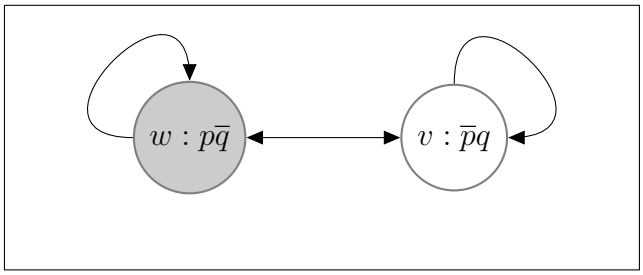
And reading formulas back into language, the formula

$$\Box_m(\neg\Box_j p \wedge \neg\Box_j \neg p)$$

says that Mary knows that John does not know if she has taken the car.

Example 5.9 (Knowledge and Negation) Note the difference between $\Box_j \neg p$ and $\neg\Box_j p$. The first expresses (under the above reading of p) that (Mary did not take the car and) John knows that Mary did not take the car, the second that (even if Mary took the car) John does not know that she took the car.

Example 5.10 (Knowledge and Disjunction) Knowing a disjunction is different from either knowing one disjunct or knowing the other disjunct. $\Box(p \vee q)$ is true in the actual world of the following model:



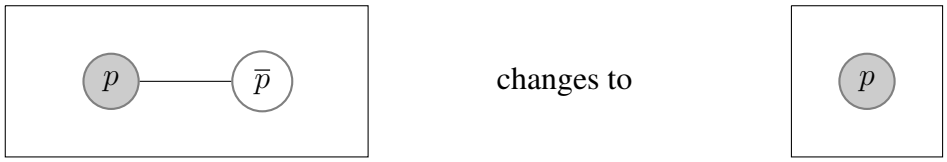
But $\Box p \vee \Box q$ is *not* true in the actual world of this model, for $\Box p$ and $\Box q$ are both false in the actual world.

Natural language has a whole range of expressions for information. We know that certain things are true, but we also know objects like telephone numbers, or methods for doing something. And other than knowing facts, we can also just "believe" them (a weaker informational notion), or we can merely consider them possible, doubt them, and so on. And these expressions are associated with actions that have names, too. You come to know facts by *learning* about them, an action that changes your information state, or as a teacher, you can *tell* other people what you know, another dynamic action. We are all experts in this informational repertoire, and it is hard to imagine life without it.

5.3 Modeling Information Change

What has been in the background of our examples all the time is that our information is not static, but that it typically keeps changing. How does this change happen? The basic idea is very simple: more information means reduction of uncertainty. And this reduction is achieved by means of shrinking the range of options, as we will see in the following pictures.

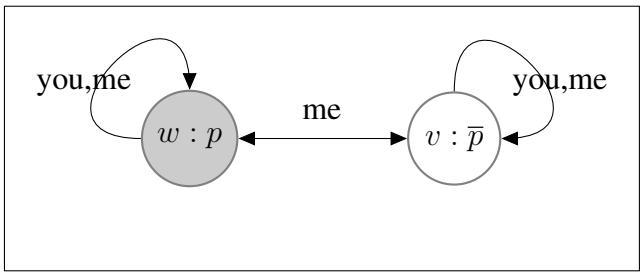
Example 5.11 (Finding out) I am uncertain whether it is raining (p) or not. I go to the window, look outside, and see that it in fact is raining. Here is what happens then to the earlier model:



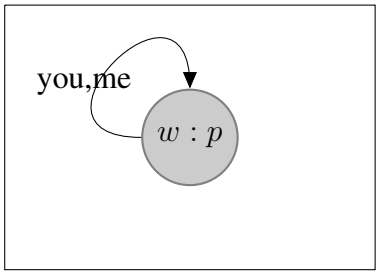
In the new smaller model, I know that p. The step taking me from the first model to the second is often called an *update*.

The same mechanism applies to acts of communication.

Example 5.12 (Answering a question.) Suppose that I do not know if p, but you do. I ask you the question "Is p the case?", and you answer truthfully. What happens when your answer? This time, the earlier model



changes to



where we both know that p. Actually, intuitively, we also know that we both know that p, since we are both in the very same situation. For more about this claim of information about other people's information, see below.

Our final example is the earlier simple card game, whose initial model was drawn above. Now the information flow already gets a bit less straightforward.

Example 5.13 (Finding out about Cards; compare Example 5.6) Suppose that the following two conversational moves take place between players:

(1) 2 asks 1: "Do you have the blue card?".

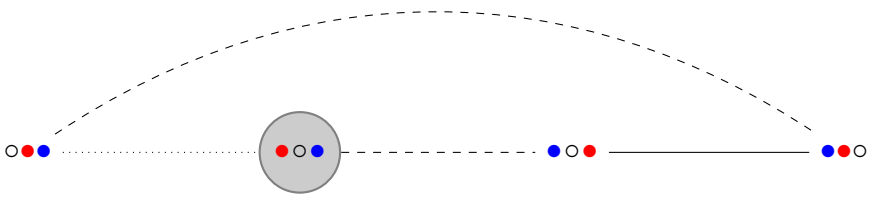
(2) 1 answers truthfully: "No".

Who knows what then? Here is the effect in words:

> Assuming the question is sincere, 2 indicates (just by asking it) that she does not know the answer, and so she cannot have the blue card. This tells 1 at once what the deal was. But 3 does not learn anything from 2's question, since he already knew that 2 does not have blue. When 1 says she does not have blue, this now tells 2 the deal. However, 3 still does not know the deal even then.

We now give the intuitive updates in the diagram, making the reasoning geometrically transparent. Here is a concrete update video of the successive information states:

After 2's question, the two situations where 2 has the blue card are removed, for the question reveals to everybody that 2 does *not* have the blue card.



After 1's answer "no", the situations where 1 has the blue card get removed, and we get:

We see at once in the final diagram that players 1, 2 know the initial deal now, as they have no uncertainty lines left. But player 3 still does not know, but she does know that 1, 2 know, and 1 and 2 know that 3 knows that 1 and 2 know, and 3 knows that 1 and 2 know that 3 knows that 1 and 2 know, and so on. In fact, that 3 knows that 1 and 2 know is *common knowledge* between the three agents: see Example 5.18 below for more information.

Note how the flow of information can be of different sorts: directly about the facts, or about what others know. Agent 3 did not find out what the deal is, but he did learn something that can also be of great interest, namely that the other players know the deal.

Exercise 5.14 (Misleading Questions.) In the preceding version of the scenario, we assumed that the question was "honest", and not misleading. That is why it gave everyone the reliable information that 2 did not have the blue card. Give examples of scenarios where questions do not indicate that the questioner does not know the answer. What if we drop this assumption in our game? What information flows then in the above scenario? What is the final diagram, and what do players know there?

Similar analyses of information flow exist for a wide variety of puzzles and games. Not all update steps are always of the above simple kind, however. We will briefly discuss more "private" forms of information flow in Outlook Section 5.10 to this chapter.

By now, we have raised a lot of issues about information and update in an informal manner. It is time to present a logical system that can deal with all this.

5.4 The Language of Epistemic Logic

We will now present the system of epistemic logic. What is important for you to see is that, even though the motivation given in this chapter may have sounded very different from that for propositional and predicate logic, the system that follows actually has a very similar technical structure. You will encounter all the topics that you have seen before: formal language, semantics, validity, proof, and update. Indeed, we will even be a bit shorter than in earlier chapters, since you can apply the techniques that you already know from there.

We start with defining the above notation more precisely.

Definition 5.15 (Basic epistemic language EL) Fix a set P of proposition letters, and a set I of agents. The basic epistemic language EL extends the language of propositional logic with modal operators $\Box_i\varphi$ ('i knows that φ'), for each agent $i \in I$. The inductive syntax rule is as follows, where p stands for any choice of proposition letters and i stands for any agent.

$$\varphi \quad ::= \quad p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid \Box_i\varphi \mid \Diamond_i\varphi.$$

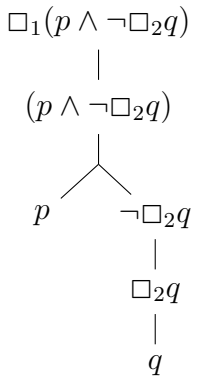
Notice that this definition covers all operations from propositional logic. We do not 'officially' put $\rightarrow$ and $\leftrightarrow$ into the language, but these can easily be defined. We know from propositional logic that $\varphi \rightarrow \psi$ can be expressed equivalently as $\neg(\varphi \wedge \neg\psi)$, and that $\varphi \leftrightarrow \psi$ can be defined as $(\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$. Henceforth we will use these abbreviations without further comment.

$\Box_i$ expresses knowledge for agent i, and $\Diamond_i$ expresses 'not knowing that not'. As the truth definition (see 5.24 below) will make clear, $\Diamond_i\varphi$ is equivalent to $\neg\Box_i\neg\varphi$. This says intuitively that 'agent i considers φ possible', or that 'according to i, φ cannot be ruled out'. The operator $\Diamond_i$ is called the *dual* of the operator $\Box_i$, just like the existential quantifier $\exists$ is the dual of the universal quantifier $\forall$.

Example 5.16 The following expression is a formula:

$$\Box_1(p \wedge \neg\Box_2 q).$$

Here is a construction tree in the same style as the ones you have seen in Chapters 2 and 4, showing the construction of the formula by following the above syntax rules step by step:



The following expressions are not formulas: $p\Box q$, $p\Box_1 q$, $\Box pq$.

Exercise 5.17 How many different correct formulas can you make from the following sequence of symbols by putting in brackets at appropriate places?

$$\neg\Box_i p \rightarrow q.$$

Also write the corresponding analysis trees for these formulas. Can you also explain what these different formulas say?

We have already shown informally how epistemic formulas match concrete natural language expressions. Here is a more elaborate example showing how this works:

Example 5.18 (Questions and Answers) I approach you in Amsterdam, and ask "Is this building the Rijksmuseum?". As a helpful Dutch citizen, you answer truly: "Yes". This is the sort of thing we all do all the time. But subtle information flows. By asking the question, I convey to you that I do not know the answer, and also, that I think it is possible that you do know. This information flows before you have said anything at all. After that, by answering, you do not just convey the topographical fact to me that this building is the Rijksmuseum. You also make me know that you know, and that you know that I know you know, etcetera. Even such a simple episode of a question followed by an answer mixes factual information with social information about the information of others. The latter type of information is not a mere "side-effect" of communication: it can steer further concrete actions. If you know that I know this building is the Rijksmuseum, and you see me running into that building waving a spray can, you may want to take some fast action.

Here is how our language can formulate some of the relevant assertions:

(i) $\neg\Box_Q\varphi \wedge \neg\Box_Q\neg\varphi$ 'questioner Q does not know whether φ',

(ii) $\Diamond_Q(\Box_A\varphi \vee \Box_A\neg\varphi)$ 'Q thinks it is possible that A knows the answer'.

After the whole two-step communication episode, φ is known to both agents:

(iii) $\Box_A\varphi \wedge \Box_Q\varphi$,

while they also know this about each other:

(iv) $\Box_Q\Box_A\varphi \wedge \Box_A\Box_Q\varphi$, $\Box_A\Box_Q\Box_A\varphi \wedge \Box_Q\Box_A\Box_Q\varphi$, etcetera.

This mutual knowledge (knowledge about knowledge of the other) to every finite depth of iteration is a property of the group $\{Q, A\}$ of the two agents together. It is called *common knowledge*.

We will return to group knowledge that arises from communication in the Outlook Section 5.11 at the end of this chapter.

Exercise 5.19 Give a concrete setting where questions have neither of the two forms of information that we mentioned in the preceding example.

Just as you have learnt with predicate logic, moving back and forth between natural language and formulas is something that you can practice.

Exercise 5.20 Write formulas that match the following sentences:

(1) John knows that it is not raining.

(2) John knows whether Mary knows that is is raining.

(3) John knows whether Mary knows if is is raining.

Exercise 5.21 Read the following formulas as sentences in natural language (pick a suitable key):

(1) $\Box_1(p \to \neg\Box_2 q)$,

(2) $\Box_1\Box_2 p \to \Box_2\Box_1 p$.

5.5 Models and Semantics for Epistemic Logic

Information models Now we state the formal version of the intuitive idea of information as range in our informal explanations. The following structures consist of a range W of 'worlds standing for all the options that we consider, while the earlier idea of lines for uncertainty is now stated formally as the presence of so-called 'accessibility relations' marked for the agents. A relation $w \to_i v$ (an arrow marked with agent i pointing from w to v) holds between two worlds if,

from the viewpoint of world w, agent i considers v a possible alternative.

Moreover, for each world, we mark which atomic facts are true there using a 'valuation'.

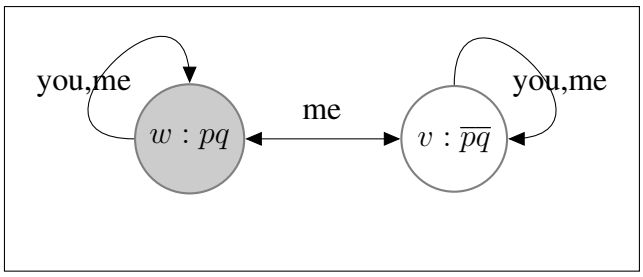
Definition 5.22 (Information Models) Models M for the epistemic language are triples

$$(W, \{\to_i |\ i \in I\}, V),$$

where W is a set of worlds, the $\to_i$ are binary accessibility relations between worlds, and V is a valuation assigning truth values to proposition letters at worlds. In what follows, we mostly use pointed models (M, s) where s is the actual world representing the true state of affairs.

Epistemic models may be viewed as collective information states for a group of agents. For illustrations, just think of all the examples we have seen already.

Example 5.23 (A Diagram Viewed as a Model) Here is a diagram that might describe the start of a question and answer episode (compare Example 5.4).



This is the following model $M = (W, \{\rightarrow_{\text{me}}, \rightarrow_{you}\}, V)$. The set W of worlds equals $\{w, v\}$, the valuation V is given by $V(w)(p) = 1$, $V(w)(q) = 1$, $V(v)(p) = 0$, $V(v)(q) = 0$, the relation $\rightarrow_{\text{me}}$ equals $\{(w, w), (w, v), (v, w), (v, v)\}$, and the relation $\rightarrow_{\text{you}}$ equals $\{(w, w), (v, v)\}$. As a pointed model, it has the shape (M, w), indicating that w is the actual world.

One way of thinking about what is happening here is that a model is not just one valuation for atomic facts, as in propositional logic, but a family of these. (This is not quite true, as we have seen in an earlier example, where different worlds could still have the same valuation for atomic facts. But often, the analogy is close enough.) Knowledge statements then refer, not just to one valuation, but to a whole range of them, as many as the agents' information requires. The case of just one world with its valuation then corresponds, as you have seen earlier, to perfect information by everybody concerning the actual world.

In general, we allow every sort of binary accessibility relations on epistemic models. Thus, any directed graph with a family of relations (indexed for the agents) could be used. We will give such a general picture in a moment. However, in practice, we often work with very special relations, of a sort already mentioned in Chapter 4 on predicate logic. These are so-called *equivalence relations*, satisfying the following three conditions:

reflexivity For all w, Rww.

symmetry For all w, v: if Rwv, then Rvw.

transitivity For all w, v, u, if Rwv and Rvu, then Rwu.

One can think of such relations as partitioning the total set of worlds into a number of disjoint maximal 'zones' of worlds connected by the relation. For instance, in the above example, the partition for Me has just one zone: the whole set $\{w, v\}$, while that for You has two zones: $\{w\}$ and $\{v\}$. This is easy to visualize, and we will soon discuss what this special structure of equivalence relations means for the logic of knowledge.

Information models arise in many settings. They are used in philosophy as a representation of what thinking human agents know. Independently, they have been proposed in economics as a representation of what players know in the course of the game: the different worlds are then different stages in a play of the game, or even different "strategy profiles" describing precisely what each player is going to do throughout the game. Chapter 7 wil have more details on connections between logic and games. But information models have also been used for describing non-human agents in computer science, say, in describing different processors in a message-passing system for communication. There is a whole area called 'Agency' where epistemic logic plays a conspicuous role.

Semantics But first, we state how the epistemic language can be interpreted on our models. The format for this is like the truth definition that you have seen for the language

of predicate logic. We start by explaining when an atomic formula is true at a world, and then work our way up along all the constructions that build formulas:

Definition 5.24 (Truth Conditions for EL)

$$\begin{array}{rll} M, s \models p & \text{iff} & V \text{ makes } p \text{ true at } s \\ M, s \models \neg\varphi & \text{iff} & \text{not } M, s \models \varphi \\ M, s \models \varphi \vee \psi & \text{iff} & M, s \models \varphi \text{ or } M, s \models \psi \\ M, s \models \varphi \wedge \psi & \text{iff} & M, s \models \varphi \text{ and } M, s \models \psi \\ M, s \models \Box_i\varphi & \text{iff} & \text{for all } t \text{ with } s \to_i t\text{: } M, t \models \varphi \\ M, s \models \Diamond_i\varphi & \text{iff} & \text{for some } t \text{ with } s \to_i t \text{ it holds that } M, t \models \varphi. \end{array}$$

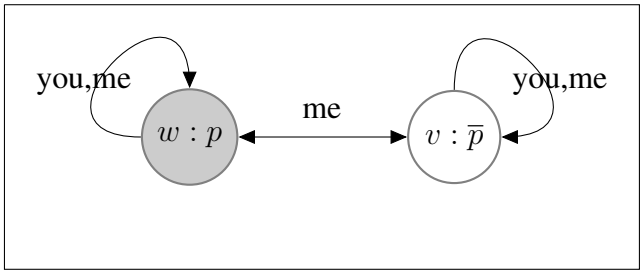
Here the clauses for the Boolean operations are exactly like those for propositional logic. The key clause is that for the knowledge operator $\Box_i\varphi$, which is read - and many people find this a helpful analogy - as a universal quantifier saying that φ is true in all accessible worlds. The epistemic operator $\Diamond_i\varphi$ can then be read dually as an existential quantifier saying that some accessible world exists satisfying φ. As was mentioned before, $\Diamond_i\varphi$ is equivalent to $\neg\Box_i\neg\varphi$. We will write

$$\Diamond\varphi$$

for this as an epistemic operator on its own, in cases where there is just a single agent (an intuitive reading would be that "φ is possible").

How does this work? Let us look at some examples.

Example 5.25 (Question and Answer Once More) Recall the above model:



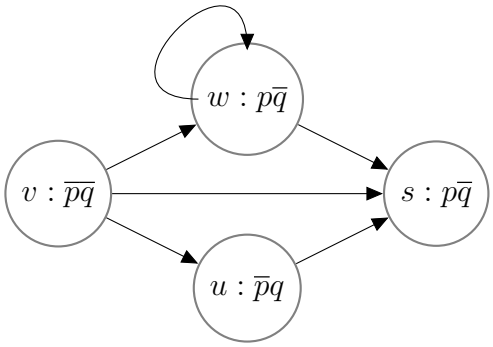
Intuitively, in the actual world (the shaded circle), I do not know whether p, but I know that you are fully informed about it. Spelling out the the above truth definition, one can check that this is right: in the actual world, $\neg\Box_{\text{me}}p \wedge \Box_{\text{me}}(\Box_{you}p \vee \Box_{you}\neg p)$ is true. For, you can see the following facts, listed in a table for convenience:

Formula	Worlds where true
p	w
$\neg p$	v
$\Box_{\text{you}} p$	w
$\Box_{\text{me}} p$	none
$\neg \Box_{\text{me}} p$	w, v
$\Box_{you} \neg p$	v
$\Box_{\text{me}} \neg p$	none
$\neg \Box_{\text{me}} \neg p$	w, v
$\neg \Box_{\text{me}} p \wedge \neg \Box_{\text{me}} \neg p$	w, v
$\Box_{you} p \vee \Box_{you} \neg p$	w, v
$\Box_{\text{me}} (\Box_{you} p \vee \Box_{you} \neg p)$	w, v.

You see the idea: evaluate simple formulas first, and find out in which worlds they are true. Then work your way upward to more complex formulas. In principle, this works no matter how large the model is, and how complex the formula you are evaluating.

Now let us also look at a much more abstract example, that does not admit an epistemic interpretation. This will allow you to see how our mechanism of interpretation works in the more general case.

Example 5.26 In the following graph, some worlds have a unique outgoing arrow to one accessible world, others have several, while there is also a world without any outgoing arrows at all: we will see in a moment what happens then.



The valuation is written into the diagram as before, marking worlds with proposition letters. Here are a few facts that you can easily check:

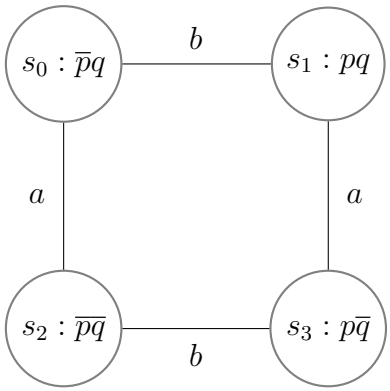
p	is true precisely in worlds	w, s
$\Box p$	is true precisely in worlds	w, u, s
$\Diamond \Box p$	is true precisely in worlds	v, w, u
$q \rightarrow p$	is true precisely in worlds	v, w, s
$\Box(q \rightarrow p)$	is true precisely in worlds	w, u, s

But we can also do something else. Each world in this model has something special, and we can describe this uniquely in the above language:

world w	is the only world satisfying	$p \wedge \Diamond p$
world v	is the only world satisfying	$\neg p \wedge \neg q$
world u	is the only world satisfying	q
world s	is the only world satisfying	$\neg\Diamond p$
world s	is the only world satisfying	$\Box\bot$.

In fact we could give many other formulas defining these four worlds uniquely. Note that $\Box\bot$ characterizes *endpoints* in the accessibility relation.

Example 5.27 (Interpreted Systems) An interpreted system is a process viewed as an information model. A process is a system that can move through various states; you will learn more about processes in Chapter 6. Our example consists of two sub-processes a and b. Propositional variable p describes the state of process a. If p is true, the state of a is 1, and if p is false the state of a is 0. Propositional variable q describes the state of process b, in a similar way. Both processes only know their own state. This corresponds to the following model.



As usual, a link labelled with a or b means that the connected states are indistinguishable for a or b, respectively. Call the model M. Now verify that $M, s_1 \models \Box_b q$ and that $M, s_3 \models \Box_a p$.

Exercise 5.28 Consider the process model from Example 5.27 again. We can describe that a knows its own state p by means of

$$(p \to \Box_a p) \wedge (\neg p \to \Box_a \neg p).$$

Similarly, b knows its own state q is expressed by:

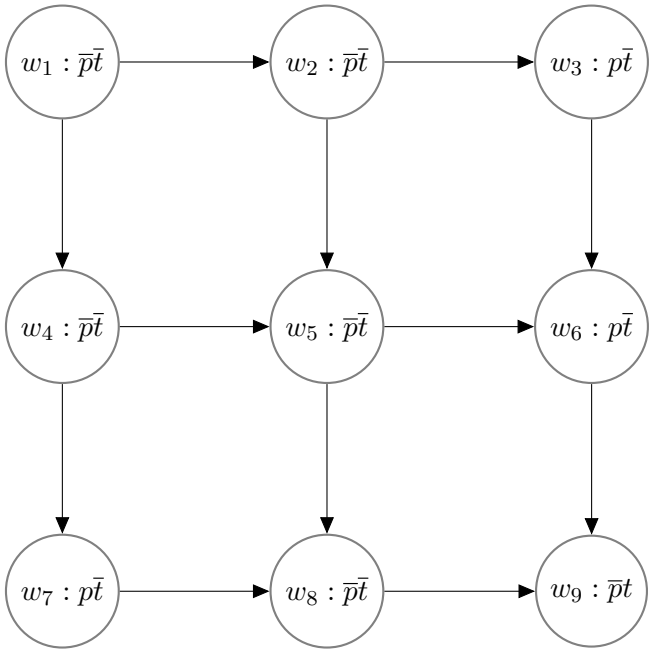
$$(q \to \Box_b q) \wedge (\neg q \to \Box_b \neg q).$$

Demonstrate that these two formulas hold in all states of the model.

Exercise 5.29 Consider the process model from Example 5.27 once more. Check that for any φ the following formula holds in all states of the model:

$$\Diamond_a \Box_b \varphi \to \Box_b \Diamond_a \varphi.$$

Exercise 5.30 (Treasure Island) Consider the following model with 9 states, and an accessibility relation allowing one step east or south (insofar as possible in the given picture) from each point. World w_9 satisfies the proposition letter t (the location of the 'treasure'), while pirates are standing at w_3, w_6, and w_7, marked by the proposition letter p.



In which worlds are the following epistemic formulas true?

(1) $\Diamond t$,

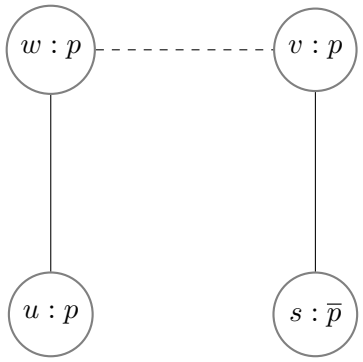
(2) $\Diamond \Box t$,

(3) $\Diamond p$,

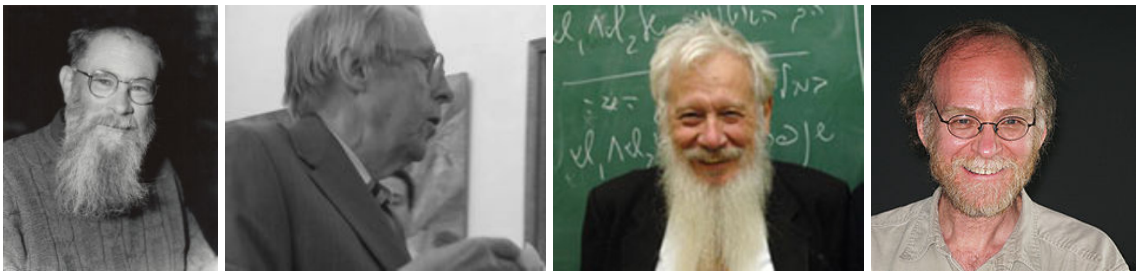
(4) $\Box \Diamond p$.

Next, for each world, find an epistemic formula which is only true at that state.

Exercise 5.31 Let's return to epistemic models in the proper sense, where the accessibility relations are all equivalences. Find epistemic formulas that uniquely define each world in the following model. The solid lines are links for agent 1, the dashed line is a link for agent 2.



Exercise 5.32 Consider the earlier models for the Three-cards scenario. Show in detail how the final model verifies the statements in the text about which player knows what about the cards, and the information of the other players.



In this picture gallery you see four people spanning the range of epistemic logic: David Lewis (a philosopher), Jaakko Hintikka (a philosopher/logician), Robert Aumann (an economist and Nobel prize winner), and Joe Halpern (a computer scientist).

5.6 Valid Consequence

We will now give definitions of valid formulas, and of valid consequence. First some examples of valid principles. We start with single operators on top of propositional logic.

$$\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi).$$

This is a kind of 'epistemic distribution': if one knows an implication then it follows that if one also knows the antecedent of the implication then one has to know the consequent too. We will adopt epistemic distribution in what follows, but we have to bear in mind that what it expresses is quite strong. It expresses the principle of so-called *logical omniscience*: our epistemic agents are perfect reasoners: they can draw the logical consequences from what they know.

$$\Diamond(\varphi \vee \psi) \leftrightarrow (\Diamond\varphi \vee \Diamond\psi).$$

Considering a disjunction possible is equivalent to holding at least one of the disjuncts for possible.

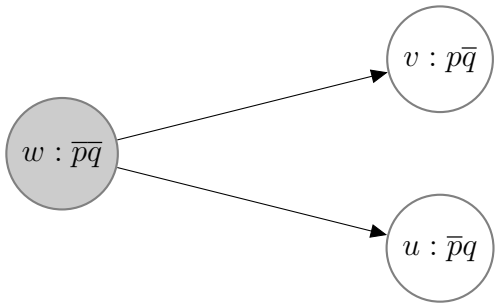
Next, there are the principles that express our abbreviation conventions:

$$\Diamond\varphi \leftrightarrow \neg\Box\neg\varphi,$$

$$\Box\varphi \leftrightarrow \neg\Diamond\neg\varphi.$$

A formula φ is called *invalid* if there is at least one model M with a world s where φ is false. Like in earlier chapters, such a pointed model (M, s) is called a counter-example for φ.

Example 5.33 (Counter-example for '$\Diamond$ over $\wedge$') In the actual world w in the following model M, $\Diamond p$ and $\Diamond q$ are true, but $\Diamond(p \wedge q)$ is not:



Here, one thing leads to another. Like in propositional and predicate logic, there are strong dualities between the two modalities and disjunction/conjunction, resulting in automatic further laws for $\Diamond$, $\Box$, $\wedge$, $\vee$. Thus, switching operators, the obvious valid counterpart to the distribution law $\Diamond(\varphi \vee \psi) \leftrightarrow (\Diamond\varphi \vee \Diamond\psi)$ is the following principle:

$$\Box(\varphi \wedge \psi) \leftrightarrow (\Box\varphi \wedge \Box\psi).$$

On the same pattern of standing and falling together, typically *invalid* principles are:

$$\Diamond(\varphi \wedge \psi) \leftrightarrow (\Diamond\varphi \wedge \Diamond\psi),$$

$$\Box(\varphi \vee \psi) \leftrightarrow (\Box\varphi \vee \Box\psi).$$

To see that $\Box(\varphi \vee \psi) \rightarrow (\Box\varphi \vee \Box\psi)$ is invalid, look at the special case where ψ equals $\neg\varphi$. As a tautology, $\varphi \vee \neg\varphi$ has to be true in any situation. Then it also has to hold that $\Box(\varphi \vee \neg\varphi)$ is true anywhere, for all tautologies are known. But it does certainly not follow that either $\Box\varphi$ or $\Box\neg\varphi$, for φ might well express some fact about which nothing is known. A concrete counterexample is the model above, where $\Box(p \vee \neg p)$ is true in world w, but both $\Box p$ and $\Box\neg p$ are false in w.

Correspondence between special relational properties and epistemic axioms Are there also interesting laws that express epistemic intuitions? That depends. First, consider one single agent. Here are three well-known axioms with prominent epistemic interpretations:

Veridicality $\Box\varphi \to \varphi$.

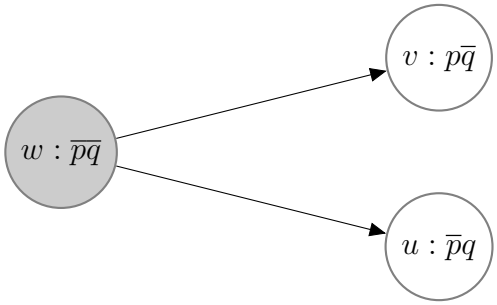
Positive Introspection $\Box\varphi \to \Box\Box\varphi$.

Negative Introspection $\neg\Box\varphi \to \Box\neg\Box\varphi$.

The first of these seems uncontroversial: knowledge has to be 'in sync' with reality, or it does not deserve to be called knowledge. If something you were firmly convinced of turns out to be false, then you may have *thought* you knew it to be true, while in actual fact you did not know it. But the other two have been much discussed. Positive introspection says that agents know what they know, and negative introspection that agents know what they do not know, and both principles seem rather strong. They seem debatable, for they assume that our epistemic agents, in addition to their logical omniscience in terms of powers of inference (encoded in the earlier distribution axiom), they now also have capacities of unlimited introspection into their own epistemic states.

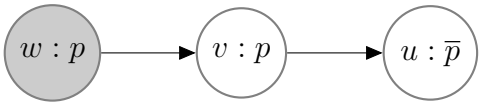
Formally, these axioms are not valid in the most general semantics that we have given.

Example 5.34 Consider the model we gave above.



Notice that in the actual world $p \vee q$ does not hold, but $\Box(p \vee q)$ does. Thus, in the actual world of the model we have $\neg(\Box(p \vee q) \to (p \vee q))$. In other words, veridicality does not hold in the model, which shows that this is not an appropriate model for representing knowledge.

Example 5.35 Consider the following model.



Note that in the actual world of the model $\Box p$ and $\neg\Box\Box p$ are both true. Thus, positive introspection does not hold in the model.

Exercise 5.36 Find a simple example of a model where $\neg\Box p \to \Box\neg\Box p$ is not true in the actual world.

It is customary to study information models with certain restrictions on their accessibility relations. Recall the "equivalence relations" that we have mentioned briefly before. If we demand that our accessibility relations in information models are of this special kind, then the above axioms become valid. In fact, the validity of each of the above axioms there is some feature in the notion of an equivalence relation that is crucially involved. The following table show this (using R for the epistemic accessibility relation):

Name	EL formula	relational principle	PL formula
Veridicality	$\Box\varphi \to \varphi$	reflexivity	$\forall x Rxx$
Pos Introsp	$\Box\varphi \to \Box\Box\varphi$	transitivity	$\forall x\forall y\forall z((Rxy \wedge Ryz) \to Rxz)$
Neg Introsp	$\neg\Box\varphi \to \Box\neg\Box\varphi$	euclidicity	$\forall x\forall y\forall z((Rxy \wedge Rxz) \to Ryz)$.

When equivalence relations were introduced on page 5-16, you saw *symmetry* instead of *euclidicity*. The following two exercises explain the connection.

Exercise 5.37 Show that every transitive and symmetric relation R is euclidic.

Exercise 5.38 Show that every euclidic and reflexive relation R is symmetric.

It follows from these two exercises that every equivalence relation is euclidic, and that every relation that is reflexive, transitive and euclidic is an equivalence relation.

To explain the connection between $\Box\varphi \to \varphi$ and reflexivity, first note that if $M, s \models \Box\varphi$ and it is given that the relation for $\Box$ is reflexive, $M, s \models \varphi$ has to hold as well, as s is among the worlds that are accessible from s. But this means that $\Box\varphi \to \varphi$ holds in all reflexive models.

To connect positive introspection and transitivity, notice that the positive introspection principle $\Box\varphi \to \Box\Box\varphi$ is equivalent to $\Diamond\Diamond\varphi \to \Diamond\varphi$ (use the definition of $\Diamond$, and do contraposition, replacing φ by $\neg\varphi$ and cancelling double negations).

Suppose $M, s \models \Diamond\Diamond\varphi$, and assume that it is given that the accessibility relation of M is transitive. Then by the truth definition there is a world t with Rst and $M, t \models \Diamond\varphi$. So, again by the truth definition, there is a world u with Rtu and $M, u \models \varphi$. Because R is transitive it follows from Rst and Rtu that Rsu, and therefore by the truth definition, $M, s \models \Diamond\varphi$. Since s was arbitrary, it follows that $\Diamond\Diamond\varphi \to \Diamond\varphi$ is valid on M.

Exercise 5.39 Which one of the following two implications is valid on models with equivalence relations? Draw a counter-example for the other:

(1) $\Diamond_1\Box_2\varphi \to \Diamond_2\Diamond_1\varphi$.

(2) $\Diamond_1\Box_2\varphi \to \Diamond_2\Box_1\varphi$.

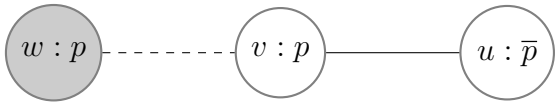
Exercise 5.40 Explain why the axiom $\neg\Box\varphi \to \Box\neg\Box\varphi$ is valid on every information model with an accessibility relation that satisfies $\forall x\forall y\forall z((Rxy \wedge Rxz) \to Ryz)$.

What about iterating knowledge of different agents? Consider $\Box_1\Box_2\varphi$. Saying that the two knowledge operators *commute* would boil down to: "I know what you know if and only if you know what I know." That does not sound plausible at all. And indeed, even when we impose the above special validities for single agents by using equivalence relations, our logical system will not validate laws that perform significant changes between knowledge of different agents. There can be interesting informative patterns on what agents know about what other agents know, to be sure, but such patterns always emerge in specific communicative contexts, as effects of what goes on in the communication. More on this below. The following example gives a counterexample to commutation of knowledge of several agents:

Example 5.41 (Your knowledge and mine do not commute) The following model is a counter-example to the putative implication

$$\Box_1\Box_2 p \to \Box_2\Box_1 p.$$

Its antecedent is true in the actual world, but its consequent is false (assume the solid line pictures the accessibility for agent 1, and the dashed line the accessibility for agent 2). Note that arrowheads and reflexive arrows are omitted from the example, since we assume that the accessibility relations are equivalences.



Such implications only hold when agents stand in special informational relationships. For example, $\Box_1\Box_2\varphi \to \Box_2\Box_1\varphi$ would hold in case it is given that $R_2 \circ R_1 \subseteq R_1 \circ R_2$.

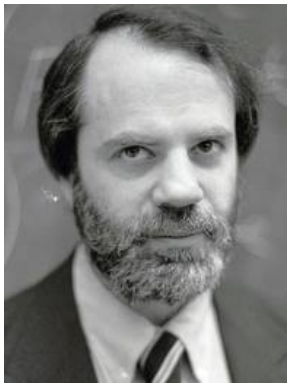
5.7 Proof

As we have done for propositional and predicate logic, in epistemic logic, too, we can establish validity by means of proof. Here are a few basic principles of epistemic reasoning in this format. Our main point is to show how the earlier style of thinking applies here too, even though our topic of information and knowledge seems quite different from that of Chapters 2 and 4. One more reason for pursuing this is that much of the pure and applied research on validity in epistemic logic, and the dynamic logic of action to follow

in Chapter 6, has focused on axiomatic proof systems, so understanding them is your key to the literature.

We will start out with a proof system for any logic with accessibilities (any so-called *modal logic*), and we will then enrich the system with special axioms describing agents with special powers for which one can derive more laws. Practically, this means that your reasoning gets richer when you know that you are dealing with agents having, e.g., the power of introspection.

The proof system we start out with is called the minimal modal logic, or the modal logic K. The name K is a historical relict; it refers to the inventor of the 'possible world' semantics for modal logic, Saul Kripke (b. 1940), and it has nothing to do with the 'K' of 'Knowledge'.



Saul Kripke

Definition 5.42 (Proof system for the minimal modal logic K) The proof system for the minimal modal logic K is given by:

(1) All propositional tautologies are theorems.

(2) All formulas of the form $\Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$ are theorems. This is called the **distribution axiom schema** or the **K axiom schema**.

(3) If φ and $\varphi \to \psi$ are theorems, then ψ is a theorem. This is the familiar **modus ponens rule**.

(4) If φ is a theorem, then $\Box\varphi$ is also a theorem. This rule is called the **necessitation rule**.

This system is an extension of the proof system for propositional logic, and it is usual, in working with this system, to perform any propositional reasoning steps without spelling out details.

We will use $\vdash \varphi$ for "φ is provable" or "φ is a theorem". E.g., $\vdash \Box p \vee \neg\Box p$, because $\Box p \vee \neg\Box p$ has the form $\varphi \vee \neg\varphi$ of a propositional tautology.

Example 5.43 (Distribution rules, 1) If $\vdash \varphi \to \psi$, then $\vdash \Box\varphi \to \Box\psi$.

(1) $\vdash \varphi \to \psi$ assumption

(2) $\vdash \Box(\varphi \to \psi)$ necessitation rule on 1

(3) $\vdash \Box(\varphi \to \psi) \to (\Box\varphi \to \Box\psi)$ distribution axiom

(4) $\vdash \Box\varphi \to \Box\psi$ modus ponens on 2, 3

Example 5.44 (Distribution rules, 2) If $\vdash \varphi \to \psi$, then $\vdash \Diamond\varphi \to \Diamond\psi$.

(1) $\vdash \varphi \to \psi$ assumption

(2) $\vdash \neg\psi \to \neg\varphi$ propositional logic, 1

(3) $\vdash \Box\neg\psi \to \Box\neg\varphi$ by subroutine of previous example

(4) $\vdash \neg\Box\neg\varphi \to \neg\Box\neg\psi$ propositional logic, 3

(5) $\vdash \Diamond\varphi \to \Diamond\psi$ definition of $\Diamond$.

Example 5.45 Formal proof of $\vdash \Box(\varphi \wedge \psi) \leftrightarrow (\Box\varphi \wedge \Box\psi)$. We will do this in two steps, proving first the implication from left to right and then the implication from right to left. Putting the two together is then just a matter of applying a propositional reasoning step.

(1) $\vdash (\varphi \wedge \psi) \to \varphi$ propositional tautology

(2) $\vdash \Box(\varphi \wedge \psi) \to \Box\varphi$ $\Box$ distribution, example 5.43

(3) $\vdash \Box(\varphi \wedge \psi) \to \Box\psi$ similarly, starting from $(\varphi \wedge \psi) \to \psi$

In propositional logic we can infer from $\vdash \varphi \to \psi$ and $\vdash \varphi \to \chi$ that $\vdash \varphi \to (\psi \wedge \chi)$. In the present system we can reason in the same way. So we get from the above:

(4) $\vdash \Box(\varphi \wedge \psi) \to (\Box\varphi \wedge \Box\psi)$.

Now for the other direction:

(5) $\vdash \varphi \to (\psi \to (\varphi \wedge \psi))$ propositional tautology

(6) $\vdash \Box(\varphi \to (\psi \to (\varphi \wedge \psi)))$ necessitation, 5

(7) $\vdash \Box\varphi \to \Box(\psi \to (\varphi \wedge \psi))$ distribution axiom, prop logic, 6

(8) $\vdash \Box\varphi \to (\Box\psi \to \Box(\varphi \wedge \psi))$ distribution axiom, prop logic, 7

(9) $\vdash (\Box\varphi \wedge \Box\psi) \to \Box(\varphi \wedge \psi)$ prop logic, 8

Finally, we use propositional logic to put 5 and 9 together, and we get:

$$\vdash \Box(\varphi \wedge \psi) \leftrightarrow (\Box\varphi \wedge \Box\psi).$$

Exercise 5.46 Prove the following:

$$\vdash (\Diamond\varphi \wedge \Box(\varphi \rightarrow \psi)) \rightarrow \Diamond\psi.$$

Stronger modal logics increase deductive power by adding further axiom schemata to the minimal logic K. Then the flavour of finding derivations may change, as you develop a feeling for what the additional power gives you. Here are some well-known examples:

Definition 5.47 (T, S4, and S5) The modal logic T arises from K by adding the axiom schema of Veridicality $\Box\varphi \rightarrow \varphi$. The logic S4 adds the schema $\Box\varphi \rightarrow \Box\Box\varphi$ to T (we have encountered this above as the principle of Positive Introspection). Finally, the logic S5 adds the schema $\Diamond\varphi \rightarrow \Box\Diamond\varphi$ to S4 (encountered above as the principle of Negative Introspection).

Example 5.48 (Alternative definition of S5) This example shows that S5 also arises from K by adding the axiom schemes of Veridicality $\Box\varphi \rightarrow \varphi$ and $\Diamond\varphi \rightarrow \Box\Diamond\varphi$ (the principle of Negative Introspection). To show that, we derive the principle of Positive Introspection from $\Box\varphi \rightarrow \varphi$ and $\Diamond\varphi \rightarrow \Box\Diamond\varphi$.

(1) $\vdash \Diamond\neg\varphi \rightarrow \Box\Diamond\neg\varphi$ negative introspection axiom schema

(2) $\vdash \Diamond\Box\varphi \rightarrow \Box\varphi$ prop logic, $\Diamond$ def, 1

(3) $\vdash \Box(\Diamond\Box\varphi \rightarrow \Box\varphi)$ necessitation, 2

(4) $\vdash \Box\Diamond\Box\varphi \rightarrow \Box\Box\varphi$ distribution, prop logic, 3

(5) $\vdash \Diamond\Box\varphi \rightarrow \Box\Diamond\Box\varphi$ instance of schema $\Diamond\varphi \rightarrow \Box\Diamond\varphi$

(6) $\vdash \Diamond\Box\varphi \rightarrow \Box\Box\varphi$ prop logic, 4, 5

(7) $\vdash \Box\Diamond\neg\varphi \rightarrow \Diamond\neg\varphi$ instance of T schema

(8) $\vdash \Box\varphi \rightarrow \Diamond\Box\varphi$ prop logic, 7

(9) $\vdash \Box\varphi \rightarrow \Box\Box\varphi$ prop logic, 8, 6.

Example 5.49 (Yet another definition of S5) The system that results by adding the schemes for veridiality, positive introspection, plus the following principle of symmetry $\varphi \rightarrow \Box\Diamond\varphi$ to K is also S5. We prove that the principle of negative introspection follows from $\varphi \rightarrow \Box\Diamond\varphi$ and $\Box\varphi \rightarrow \Box\Box\varphi$:

(1) $\vdash \Diamond\varphi \rightarrow \Box\Diamond\Diamond\varphi$ instance of $\varphi \rightarrow \Box\Diamond\varphi$

(2) $\vdash \Diamond\Diamond\varphi \rightarrow \Diamond\varphi$ prop logic, from $\Box\neg\varphi \rightarrow \Box\Box\neg\varphi$

(3) $\vdash \Box(\Diamond\Diamond\varphi \rightarrow \Diamond\varphi)$ necessitation, 2

(4) $\vdash \Box\Diamond\Diamond\varphi \rightarrow \Box\Diamond\varphi$ distribution, prop logic, 3

(5) $\vdash \Diamond\varphi \rightarrow \Box\Diamond\varphi$ prop logic, 1, 4

And here is a derivation of $\varphi \rightarrow \Box\Diamond\varphi$, using only the K schema, the T schema and the schema of negative introspection:

(1) $\vdash \Box\neg\varphi \rightarrow \neg\varphi$ instance of $\Box\varphi \rightarrow \varphi$

(2) $\vdash \varphi \rightarrow \Diamond\varphi$ def of $\Diamond$, prop logic, 1

(3) $\vdash \Diamond\varphi \rightarrow \Box\Diamond\varphi$ axiom

(4) $\vdash \varphi \rightarrow \Box\Diamond\varphi$ prop logic, 2, 3

Example 5.50 (Reduction of Modalities for S5) We will now prove that S5 allows reduction of modalities, where a modality is a list of modal operators $\Box, \Diamond$. First we show that $\Diamond\varphi \leftrightarrow \Box\Diamond\varphi$ is a theorem:

(1) $\vdash \Diamond\varphi \rightarrow \Box\Diamond\varphi$ axiom schema of Negative Introspection

(2) $\vdash \Box\Diamond\varphi \rightarrow \Diamond\varphi$ instance of T axiom schema

(3) $\vdash \Diamond\varphi \leftrightarrow \Box\Diamond\varphi$ prop logic, 1, 2

By contraposition and the interdefinability of $\Box$ and $\Diamond$ we get from this that $\Box\varphi \leftrightarrow \Diamond\Box\varphi$ is also a theorem. It is easy to show that $\Box\varphi \leftrightarrow \Box\Box\varphi$ is an S5 theorem:

(1) $\vdash \Box\Box\varphi \rightarrow \Box\varphi$ instance of T axiom schema

(2) $\vdash \Box\varphi \rightarrow \Box\Box\varphi$ positive introspection schema

(3) $\vdash \Box\varphi \leftrightarrow \Box\Box\varphi$ prop logic, 1, 2

By contraposition and the interdefinability of $\Box$ and $\Diamond$ we get from this that $\vdash \Diamond\varphi \leftrightarrow \Diamond\Diamond\varphi$.

Thus we see, that in S5, it is always the innermost operator that counts: $\Diamond\Box$ and $\Box\Box$ reduce to $\Box$, $\Box\Diamond$ and $\Diamond\Diamond$ reduce to $\Diamond$.

But note that we are talking here about a system for reasoning about a single agent. In the multi-agent logic of knowledge, $\Box_i\Box_j\varphi$ does *not* reduce to a single modality.

Exercise 5.51 Which of the following two implications is valid? Give an informal argument, and also a formal proof in the minimal logic K:

(1) $\Box(p \to q) \to (\Diamond p \to \Diamond q)$.

(2) $(\Diamond p \to \Diamond q) \to \Box(p \to q)$.

As for the invalid formula, give a counterexample (draw the model).

Exercise 5.52 Prove the formula $\Box(p \vee q) \to (\Diamond p \vee \Box q)$.

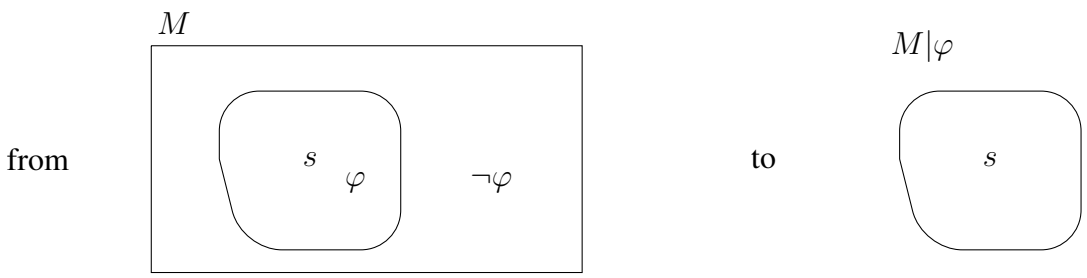
5.8 Information Update

In Section 5.3 we explained the intuitive notion of information update through successive elimination of possibilities. We will now make this more precise, and explain how information sources like observation and communication through update can be dealt with in a formal way, by describing the process by which information states are changed by incoming information, to yield new information states (or: 'updated' information states).

This update process works over pointed information models (M, s) with s the actual world. When new information arrives, we can think of this as a public announcement of some true proposition φ, where "true" means that φ holds in the actual world: $M, s \models \varphi$. 'Public announcement' is a generic technical term for various things that could be going on: it could a public broadcast, but it might just as well be some publicly observable happening like the outbreak of a thunderstorm, or a sunset witnessed by all.

Definition 5.53 (Updating via definable submodels) For any epistemic model M, world s, and formula φ true at s, the model $(M|\varphi, s)$ (M relativized to φ at s) is the sub-model of M whose domain is the set $\{t \in W_M \mid M, t \models \varphi\}$ (where W_M indicates the set of worlds of M).

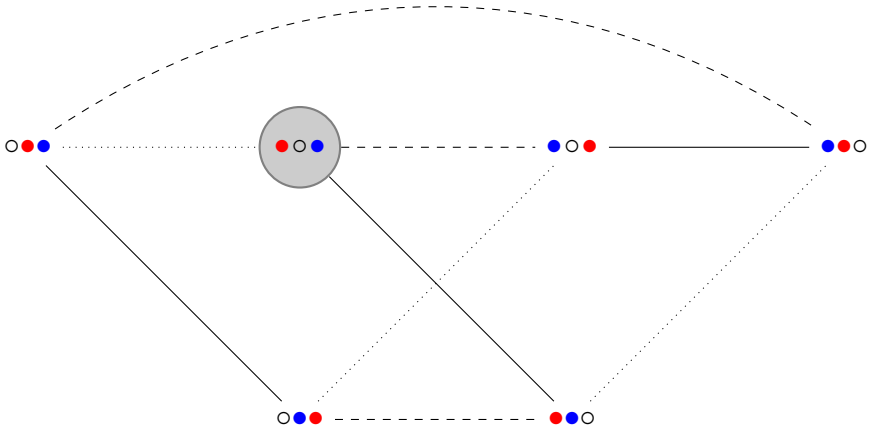
Drawn in a simple picture, such an update step goes



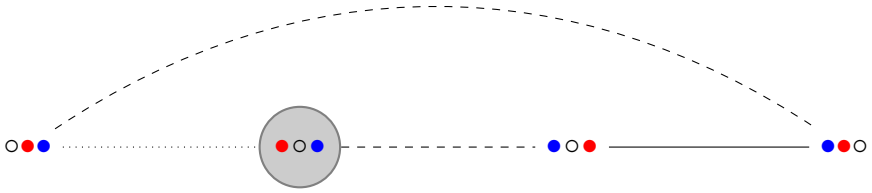
The fact that we eliminate means that the $\neg\varphi$-worlds are ruthlessly discarded: they are gone forever. This is often called 'hard information', an act which changes the current

model irrevocably as a response to some totally trusted source. You can think of this as a typical step of communication when something is said by a trusted authority, but another good reading is as an act of public observation, whether or not stated in language.

Example 5.54 We return to the Three Cards example (Example 5.6), to see how this fits this format. Here is the picture of the situation after the card deal again:



The question of agent 2 to agent 1 "Do you have the blue card", if honest, implies the statement "I know that I do not have the blue card", or, formally, $\Box_2 \neg b_2$ (if we assume a language with proposition letters b_1, b_2, b_3 with the obvious meanings). From $\Box_2 \neg b_2$ it follows that $\neg b_2$; the worlds where $\neg b_2$ is true are precisely the worlds ○●●, ●○●, ●●○ and ●●○, so the model restricted to $\neg b_2$ is indeed as we saw before:

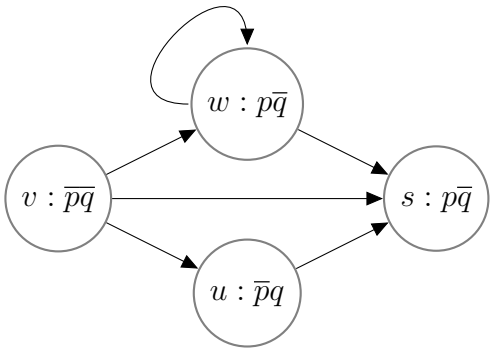


One word about the term "public": we use this, because the above elimination operation is the same for all the agents that live in the model. They all "see" the same structure afterwards. In reality, of course, the world is full of differences in observation, and even hiding of information. The information flow in scenarios with privacy is much more subtle than what we discuss here: see Outlook Section 5.10 below for a few thoughts.

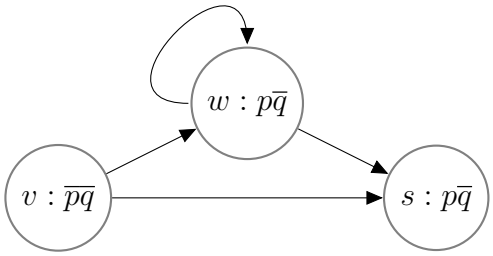
These diagrams of a jump from one model to another seem to be simple, but they are trickier than you might think. Crucially, truth values of epistemic formulas at a world may change in an update step as depicted here. For instance, when a public announcement $!p$ is made, only the p-worlds remain (we use $!p$ for the public announcement of the fact p, and

more generally, $!\varphi$ for the public announcement of φ). But at p-worlds where formerly, the agent did not know that p because of the presence of some accessible $\neg p$-world, the agent knows p after the update, since there are only p-worlds around now.

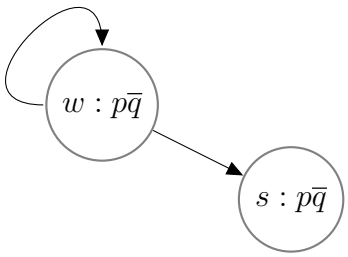
Example 5.55 Recall the model of example 5.26, repeated here:



If this model is called M, what is $M|p \vee \Diamond q$? To calculate the model, just check where $p \vee \Diamond q$ is true, and we get the set $\{v, w, s\}$ (for u is the only world where $p \vee \Diamond q$ fails. Therefore, $M|p \vee \Diamond q$ looks like this:



Now we can evaluate $p \vee \Diamond q$ once again. This time v fails the formula, because the world that made $\Diamond q$ true before is no longer present. We get:



Finally, if we update with $p \vee \Diamond q$ yet another time, nothing changes anymore.

Exercise 5.56 Find a simple example of a model M with the property that M, $M|\Diamond p$, $(M|\Diamond p)|\Diamond p$, and $((M|\Diamond p)|\Diamond p)|\Diamond p$ are all different.

This update mechanism, simple though it may seem, explains many knowledge puzzles, one of which is an evergreen, as it packs many relevant topics into one simple story.[1]

Example 5.57 (Muddy Children) Three children play outside, and two of them get mud on their foreheads. They can only see the other children's foreheads, so they do not know whether they themselves are muddy or not. (This is an inverse of our card games.) Now their father says: "At least one of you is dirty". He then asks: "Does anyone know if he is dirty?" (with 'knowing if you are dirty' the father means 'knowing that you are dirty if you are and knowing that you are not dirty if you are not'). The children answer truthfully. As questions and answers repeat, what happens? Assume that the children are all perfect reasoners, and this is commonly known among them.

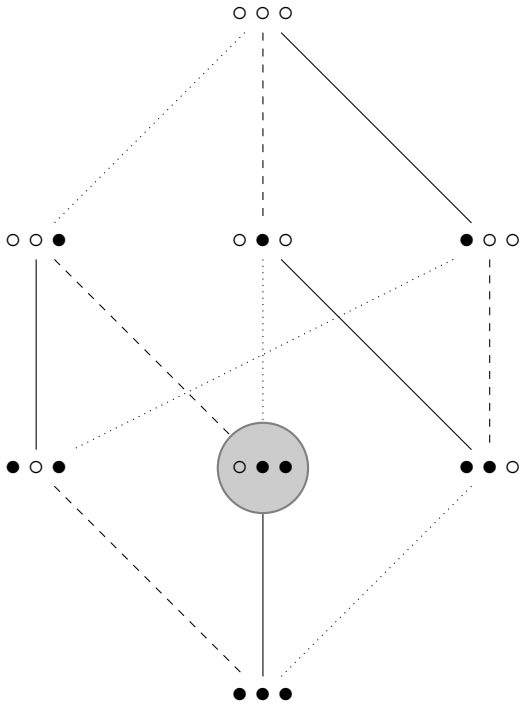
Nobody knows in the first round. But in the next round, each muddy child can reason like this: "Suppose I were clean. Then the one dirty child I see would have seen only clean children, and so she would have known that she was dirty at once. But she did not. So I must be dirty, too!"

So what happens is this:

	1	2	3
"At least one of you is dirty"	○	●	●
"Does anyone know if he is dirty?"	"No"	"No"	"No"
"Does anyone know if he is dirty?"	"No"	"Yes"	"Yes"
"Do you know if you are dirty?"	"Yes"		

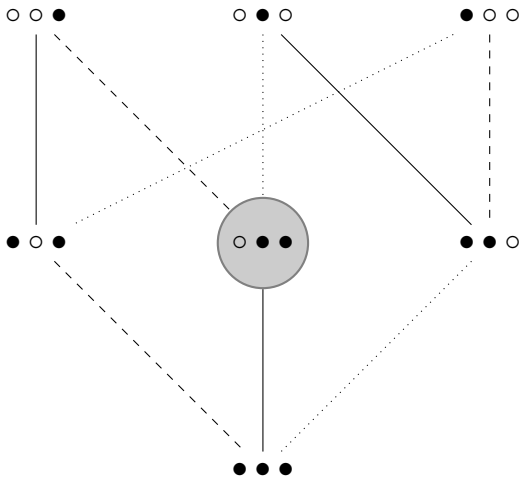
In the initial model, eight possible worlds assign "dirty" or "clean" to each child. We indicate this pictorially as ○ ● ●, for "child 1 clean, children 2 and 3 dirty" and so on. Suppose the actual situation is ○ ● ●. In the language, we assume we have proposition letters d_1, d_2, d_3, for "child 1 is dirty", "child 2 is dirty", "child 3 is dirty", respectively. So the situation ○ ● ● is described by $\neg d_1 \wedge d_2 \wedge d_3$. We can represent the initial model like this:

[1] The Muddy Children Puzzle has been doing the rounds for a long time. The currently earliest known source is a long footnote in a German 1832 translation of the French novel sequence *La vie de Gargantua et de Pantagruel* by Rabelais. This is a version with charcoal on noses, instead of mud on foreheads.



Solid lines are links for child 1, dashed lines are links for child 2, dotted lines links for child 3. There is a solid line between ○ ○ ○ and ● ○ ○, for the first child cannot distinguish the situation where the are all clean from the situation where the other two are clean and she is dirty, and so on. So the fact that the children all know about the others' faces, and not their own, is reflected in the accessibility links in the diagram.

Now let us see what happens when father makes his announcement. In a formula: $d_1 \vee d_2 \vee d_3$. The effect of this is that the world with ○ ○ ○ disappears:

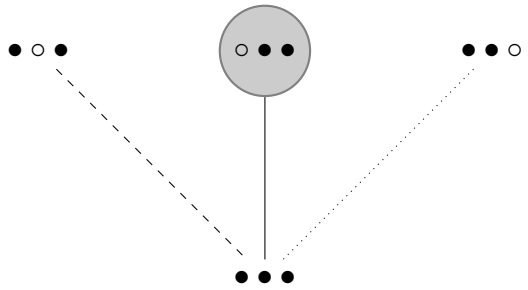


Now in response to father's first question "Does any of you know whether you are dirty?"

the children all say "No". These are three announcements:

$$\neg\Box_1 d_1 \wedge \neg\Box_1\neg d_1, \neg\Box_2 d_2 \wedge \neg\Box_2\neg d_2, \neg\Box_3 d_3 \wedge \neg\Box_3\neg d_3.$$

The first of these is false in world • ○ ○, the second is false in world ○ • ○, and the third is false in world ○ ○ •. (But note that these three formulas would all have been true everywhere in the initial model, before father's announcement.) So an update with these announcements results in the following new model:



Now father asks again, and the answers are:

$$\neg\Box_1 d_1 \wedge \neg\Box_1\neg d_1, \Box_2 d_2 \vee \Box_2\neg d_2, \Box_3 d_3 \vee \Box_3\neg d_3.$$

These answers make • ○ •, • • ○ and • • • disappear, and the final update results in a situation where only the actual world remains:

In this sequence of models, domains decrease in size stepwise: from 8 worlds to 7, then to 4, then down to 1. With k muddy children, k rounds of the simultaneous assertion "I do not know my status" yield common knowledge of which children are dirty. A few additional assertions by those who now know are then enough to establish common knowledge of the complete distribution of the mud on the faces of the group.

Exercise 5.58 Suppose that in the Three-Card example, the question of Player 2 is not treated as informative. What happens then? Draw the two updates.

Exercise 5.59 Suppose that in our Muddy Children scenario with 3 kids, the children speak in turn, each time starting from the first child, then the second, and finally the third. What happens? Draw the successive diagrams.

Exercise 5.60 Suppose in the Muddy Children scenario, with 3 children, the father says "At least one of you is clean", and then the same procedure is followed as before. Compute the update sequence, and explain what happens.

Exercise 5.61 Three men are standing on a ladder, each wearing a hat. Each can see the colours of the hats of the people below him, but not his own or those higher up. It is common knowledge that only the colours red and white occur, and that there are more white hats than red ones. The actual order is white, red, white from top to bottom. Draw the information model. The top person says: "I know the color of my hat". Is that true? Draw the update. Who else knows his color now? If that person announces that he knows his colour, what does the bottom person learn?

We end with some examples with a more philosophical flavour. One of the methods that philosophers have developed to clear up conceptual muddles is the use of bits of formal proofs to make their arguments precise. This is one more motivation for us to teach you at least some basic skills in proof reading, and proof search.

Example 5.62 (Moore-type sentences) Public announcement of atomic facts p (or more generally, purely propositional facts) makes them common knowledge. But not all events $!\varphi$ result in common knowledge of φ. A counter-example are so-called 'Moore-type' sentences. In a question-answer scenario, let the answerer A say truly

$$p \wedge \neg \Box_Q p \quad \text{"}p\text{, but you don't know it"}$$

This removes Qs ignorance about p, and thus makes itself false: true sentences like this lead to knowledge of their negation. This also occurred with the Muddy Children, where the last assertion of ignorance led to knowledge.

Example 5.63 (Verificationism and the Fitch paradox) The general verificationist thesis (VT) says that what is true can be known – or formally:

$$\varphi \to \Diamond \Box \varphi. \tag{VT}$$

A surprising argument by the philosopher Frederic Fitch (1908–1987) trivializes this principle, taking the substitution instance

$$(\varphi \wedge \neg \Box \varphi) \to \Diamond \Box (\varphi \wedge \neg \Box \varphi).$$

Then we have the following chain of three conditionals (say, in the weak modal logic T):

(1) $\Diamond \Box (\varphi \wedge \neg \Box \varphi) \to \Diamond (\Box \varphi \wedge \Box \neg \Box \varphi)$

(2) $\Diamond (\Box \varphi \wedge \Box \neg \Box \varphi) \to \Diamond (\Box \varphi \wedge \neg \Box \varphi)$

(3) $\Diamond (\Box \varphi \wedge \neg \Box \varphi) \to \bot$.

Here, $\bot$ is shorthand for a formula that is always false, say $p \wedge \neg p$.

Thus, a contradiction has been derived from the assumption $\varphi \wedge \neg \Box \varphi$, and we have shown over-all that φ implies $\Box \varphi$, making truth and knowledge equivalent. Here it seems plausible to read the modality as an event of getting hard information, and then the point is again that the Moore sentence $\varphi \wedge \neg \Box \varphi$ cannot be truly announced without making itself false.

5.9 The Logic of Public Announcement

In Chapters 3, 2, 4 we did not have to worry about the subtleties of information change, since each model pictured just one single unchanging setting. Now we have added change, and a logical system helps us be precise and consistent about reasoning in the presence of change. To do so, we must bring the informational actions themselves explicitly into the logic.

Language A suitable language for this is a combination of epistemic and dynamic logic. Dynamic logic is a tool for studying action – much more about this in Chapter 6 – and languages for dynamic logic include action expressions. We will focus here on the action of making public announcements.

Definition 5.64 (Language and semantics of public announcement) The language of public announcement logic PAL (without common knowledge) is the epistemic language with added action expressions, as well as dynamic modalities for these, defined by the syntax rules:

$$\begin{array}{rrl} \text{Formulas } \varphi & ::= & p \mid \neg\varphi \mid (\varphi \vee \varphi) \mid (\varphi \wedge \varphi) \mid \Box_i\varphi \mid \Diamond_i\varphi \mid [A]\varphi \mid \langle A\rangle\varphi \\ \text{Action expressions } A & ::= & !\varphi \end{array}$$

In Outlook Section 5.11 we will extend this language with an operator for expressing common knowledge.

Note the mutual recursion in this definition: action expressions contain formulas which may contain action expressions which contain formulas, and so on. There is no harm in this, for each embedded action is syntactically less complex than the expression it is a part of, and the recursion comes to an end.

Semantics The epistemic language is interpreted as before in Section 5.5, while the semantic clause for the new dynamic action modality is 'forward-looking' among models as follows:

$$\begin{array}{lll} M, s \models [!\varphi]\psi & \text{iff} & M, s \models \varphi \text{ implies } M|\varphi, s \models \psi \\ M, s \models \langle !\varphi\rangle\psi & \text{iff} & M, s \models \varphi \text{ and } M|\varphi, s \models \psi. \end{array}$$

This language can make typical assertions about knowledge change like $[!\varphi]\Box_i\psi$, which states what an agent i will know after having received the hard information that φ.

Example 5.65 We return again to example 5.6 (and 5.54). We evaluate the following formula in the model of the initial situation, just after the card deal:

$$[!\neg b_2][!\neg b_1]\Box_1(\neg K_3 w_1 \wedge \neg K_3 r_1).$$

What this says is: after the public announcement "2 does not have blue", followed by the public announcement update with "1 does not have blue", player 1 knows that player 3 does not know 1's card. To compute whether this is true, just perform the updates and evaluate $\Box_1(\neg K_3 w_1 \wedge \neg K_3 r_1)$ in the actual world of the resulting model:

The formula is true in the actual world. So the original formula is true in the actual world of the model we started out with.

Exploring updates Call a public announcement $!\varphi$ *factual* if φ is a purely propositional formula (no modality $\Box_i$ or $\Diamond_i$ occurs in φ). Suppose φ is true in the actual world. Then an announcement of φ makes φ known to all agents. In fact, φ becomes common knowledge as a result of its announcement. For epistemic formulas this need not be the case. Indeed, we have seen some examples of that already: Moore-type formulas, but also the public announcements of ignorance in the Muddy Children scenario, where announcing ignorance may create knowledge. A formula like $\neg\Box p$ may become known after being announced, but it need not be. See the next exercise.

Exercise 5.66 Find a pointed model (M, s) where $M|\neg\Box p \models \neg\Box p$, and a pointed model (N, t) where $N|\neg\Box p \models \Box p$.

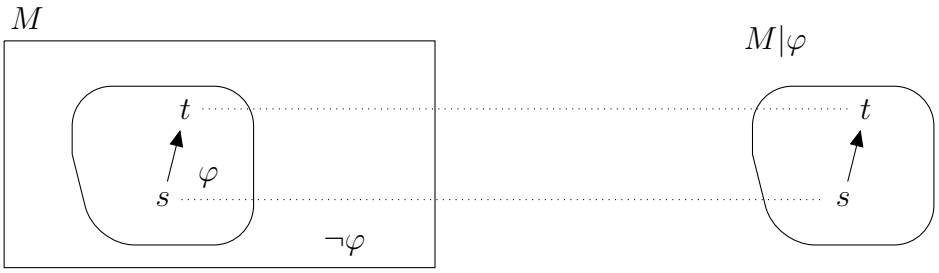
And we have seen that a Moore sentence like $(p \wedge \neg\Box p)$ always makes itself false when announced. Epistemic formulas that make themselves false when announced are not oddities but can be very useful: see the ignorance assertions in Muddy Children above, that led to knowledge.

Valid reasoning and proof Reasoning about information flow in public update satisfies axioms, just as we saw before with reasoning about knowledge and information at some fixed stage, i.e., some given information model. This reasoning revolves around the following dynamic 'recursion equation', which relates new knowledge to old knowledge the agent had before:

Valid PAL equivalence, useful for axiomatizing announcements The following equivalence is valid for PAL:

$$[!\varphi]\Box_i\psi \leftrightarrow (\varphi \rightarrow \Box_i(\varphi \rightarrow [!\varphi]\psi)).$$

To see why this is true, compare the two models (M, s) and $(M|\varphi, s)$ before and after the update.



The formula $[!\varphi]\Box_i\psi$ says that, in $M|\varphi$, all worlds t that are i-accessible from s satisfy ψ. The corresponding worlds t in M are those i-accessible from s which satisfy φ. As truth values of formulas may change in an update step, the right description of these worlds in M is not that they satisfy ψ (which they do in $M|\varphi$), but rather $[!\varphi]\psi$: they become ψ after the update. Finally, $!\varphi$ is a partial function (a function that is not everywhere defined), for φ must be true for its announcement to be executable (and if the announcement is not executable, the update result is undefined). Thus, we make our assertion on the right (the assertion about the model after the update) conditional on $!\varphi$ being executable, i.e., on φ being true. Putting this together, $[!\varphi]\Box_i\psi$ says the same as $\varphi \to \Box_i(\varphi \to [!\varphi]\psi)$.

Here is how this functions in a complete calculus of public announcement (we state the theorem without proof):

Theorem 5.67 PAL is axiomatized completely by the laws of epistemic logic over our static model class plus the following recursion axioms:

$$\begin{aligned}
[!\varphi]p &\leftrightarrow \varphi \to p \text{ for atomic facts } p\\
[!\varphi]\neg\psi &\leftrightarrow \varphi \to \neg[!\varphi]\psi\\
[!\varphi](\psi \wedge \chi) &\leftrightarrow [!\varphi]\psi \wedge [!\varphi]\chi\\
[!\varphi]\Box_i\psi &\leftrightarrow \varphi \to \Box_i(\varphi \to [!\varphi]\psi).
\end{aligned}$$

Note that these axioms are all equivalences. To reason with such equivalences we can use the following principle:

Leibniz' principle: If $\vdash \varphi \leftrightarrow \psi$ and χ' is the result of replacing a subformula φ in χ by ψ, then $\vdash \chi \leftrightarrow \chi'$.

This principle is used several times in the following example. One application is the inference from $\vdash [!q]q \leftrightarrow (q \to q)$ to $\vdash (q \to \Box(q \to [!q]q)) \leftrightarrow (q \to \Box(q \to (q \to q)))$.

Example 5.68 (Announcing an atomic fact makes it known) Here is a typical calculation using the axioms (we use $\top$ for a formula that is always true, say $p \vee \neg p$).

$$\begin{aligned}
[!q]\Box q &\leftrightarrow (q \to \Box(q \to [!q]q))\\
&\leftrightarrow (q \to \Box(q \to (q \to q)))\\
&\leftrightarrow (q \to \Box\top)\\
&\leftrightarrow (q \to \top)\\
&\leftrightarrow \top.
\end{aligned}$$

The second step uses the equivalence of $[!q]q$ and $q \to (q \to q)$, the third that of $q \to (q \to q)$ and $\top$, the fourth that of $\Box\top$ and $\top$. To see that $\vdash \Box\top \leftrightarrow \top$, notice that $\vdash \Box\top \to \top$ is an instance of the **T** axiom schema, while from $\vdash \top$ we get $\vdash \Box\top$ by necessitation, and from $\vdash \Box\top$ we get $\vdash \top \to \Box\top$ by propositional logic.

Example 5.69 (Announcement of propositional facts is order-independent)

$$\begin{array}{rcl} [!p][!q]\varphi & \leftrightarrow & [!p](q \to \varphi) \\ & \leftrightarrow & (p \to (q \to \varphi)) \\ & \leftrightarrow & ((p \wedge q) \to \varphi). \end{array}$$

Example 5.70 (Moore sentences again) Let us calculate the conditions under which Moore announcements do make themselves true, using the axioms. First we do a separate calculation to compute $[!(p \wedge \neg\Box p)]\Box p$:

$$\begin{array}{rcl} [!(p \wedge \neg\Box p)]\Box p & \leftrightarrow & (p \wedge \neg\Box p) \to \Box((p \wedge \neg\Box p) \to [!(p \wedge \neg\Box p)]p) \\ & \leftrightarrow & (p \wedge \neg\Box p) \to \Box\top \\ & \leftrightarrow & (p \wedge \neg\Box p) \to \top \\ & \leftrightarrow & \top. \end{array}$$

Next, we are going to use this:

$$\begin{array}{rcl} [!(p \wedge \neg\Box p)](p \wedge \neg\Box p) & \leftrightarrow & [!(p \wedge \neg\Box p)]p \wedge [!(p \wedge \neg\Box p)]\neg\Box p \\ & \leftrightarrow & ((p \wedge \neg\Box p) \to p) \wedge ((p \wedge \neg\Box p) \to \neg[!(p \wedge \neg\Box p)]\Box p) \\ & \leftrightarrow & ((p \wedge \neg\Box p) \to \neg[!(p \wedge \neg\Box p)]\Box p) \\ & \leftrightarrow & ((p \wedge \neg\Box p) \to \bot \\ & \leftrightarrow & \neg p \vee \Box p. \end{array}$$

In the next-to-last line of this derivation we used the fact we proved before: that $[!(p \wedge \neg\Box p)]\Box p \leftrightarrow \top$ is a theorem, and therefore, that $\neg[!(p \wedge \neg\Box p)]\Box p \leftrightarrow \bot$ is a theorem too.

What this derivation says is that update with $!(p \wedge \neg\Box p)$ results in $p \wedge \neg\Box p$ in precisely those cases where the update *cannot be executed* because what it expresses is false.

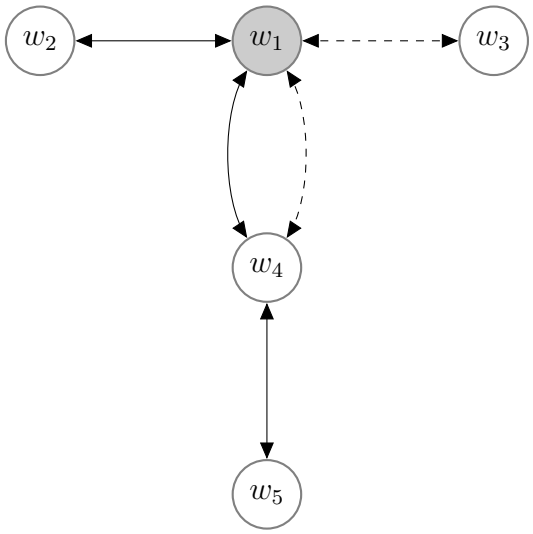
Example 5.71 (Conversation) PAL may be used as a logic of longer-term conversations, or observation procedures, by iterating single update steps. Here is a relevant observation:

Fact 5.72 The formula $[!\varphi][!\psi]\chi \leftrightarrow [!(\varphi \wedge [!\varphi]\psi)]\chi$ is valid.

This formula describes how in sequences of two announcements the second announcement is interpreted 'relative' to the update effect of the first.

Optimal communication What can agents in a group achieve by maximal communication? Consider two epistemic agents that find themselves in some collective information state M, at some actual situation s. They can tell each other things they know, thereby cutting down the model to smaller sizes. Suppose they wish to be maximally cooperative.

Example 5.73 (The best agents can do by internal communication) What is the best that can be achieved in the following model? Assume solid links are (symmetric) accessibilities for Q, and dashed links accessibilities for A. Note that in this example the accessibilities are not reflexive.



Geometrical intuition suggests that this must be:

Indeed, a two-step conversation getting here is the following:

- Q sighs: "I don't know".
- Then A sighs: "I don't know either".

It does not matter if you forget details, because it also works in the opposite order.

But maybe we have to assume the accessibilities in the example express belief rather than knowledge, because, as we have seen, knowledge models always have reflexive accessibilities. The accessibilities in the model are not reflexive. If we reinterpret the links in the example model as links expressing belief, the following conversation has the desired effect:

- Q, with indignation: "I don't believe just anything, you know".
- Then A, also indignant: "Well, neither do I".

The first update is with the formula $\neg\Box_Q\bot$, the second with $\neg\Box_A\bot$.

Exercise 5.74 Give equivalent versions for the PAL axioms with existential modalities $\langle!\varphi\rangle$, where $\langle!\varphi\rangle\psi$ is defined as $\neg[!\varphi]\neg\psi$.

A remarkable feature of the axioms for PAL is that the principles about public announcements in the axiomatisation are all equivalences. Also, on the left-hand sides the public announcement operator is the principal operator, but on the righthand sides it is not. What this means is that the axioms reveal that PAL is much more expressive than one might think. It turns out that PAL can encode intricate dynamics of information, provided you take the trouble of analyzing what goes on in information update, in the way we have done.

The principles we have uncovered (in the form of axioms for information update) can be used to 'translate' a formula of PAL to a formula of our standard epistemic language EL. In other words: every statement about the effects of public announcement on individual knowledge is equivalent to a statement about just individual knowledge.

It should be noted, however, that this reduction goes away when we look at temporal processes, protocols and games, the next area one can go from here.

5.10 Outlook — Information, Knowledge, and Belief

From knowledge to belief While information and knowledge are important, our actions are often driven by less demanding attitudes of belief. I ride my bicycle since I believe that it will get me home, even though I can imagine worlds where an earthquake happens. With this distinction in attitude comes one of dynamics. An event of hard information changes irrevocably what I know. If I see the Ace of Spades played on the table, I come to know that no one holds it any more. But there are also events of soft information, which affect my current beliefs without affecting my knowledge in a card game. I see you smile. This makes it more likely that you hold a trump card, but it does not rule out that you do not. How to model all this?

Belief and plausibility models An agent believes what is true, not in all epistemically accessible worlds, but only in the most plausible ones. I believe my bicycle will get me home early, even though I do not know that it will not disappear in an earthquake chasm. But worlds where it stays on the road are more plausible than those where it drops down, and among the former, those where it arrives on time are more plausible than those where it does not.

Definition 5.75 (Epistemic-doxastic models) Epistemic-doxastic models are structures

$$M = (W, \{\sim_i |\ i \in I\}, \{\leq_i |\ i \in I\}, V)$$

where the relations $\sim_i$ stand for epistemic accessibility, and the $\leq_i$ are comparison relations for agents read as follows:

$$x \leq_i y \text{ if agent } i \text{ considers } x \text{ at least as plausible as } y.$$

One can impose several conditions on the plausibility relations, depending on their intuitive reading. An often-used minimum is reflexivity and transitivity, while a lusher version adds

Connectedness For all worlds s, t, either $s \leq t$ or $t \leq s$.

Definition 5.76 (Belief as truth in the most plausible worlds) In epistemic-doxastic models, knowledge is interpreted as usual, while we now say that

$$M, s \models B_i \varphi$$

iff $M, t \models \varphi$ for all worlds t that are minimal in the ordering $\leq_i$.

This can be further refined, as follows.

Definition 5.77 (Conditional Belief as Plausibility Conditionals) Extend the language with conditional belief formulas $B_i^{\psi}\varphi$, with the intuitive reading that, conditional on ψ, the agent believes that φ. Formally:

$$M, s \models B_i^{\psi}\varphi \quad \text{iff} \quad M, t \models \varphi \text{ for all worlds } t \text{ which are minimal for the ordering } \leq_i \text{ in the set } \{u \mid M, u \models \psi\}.$$

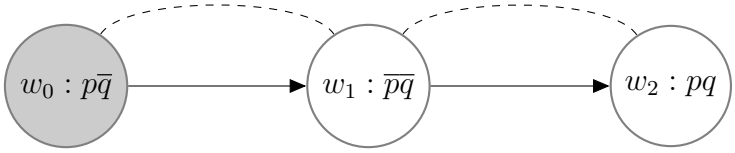
Belief change under hard information The capacity for learning from new facts contradicting our earlier beliefs seems typical of rational agency.

Fact 5.78 The formula

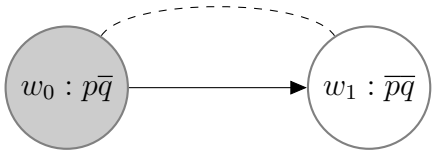
$$[!\varphi]B^{\psi}\chi \leftrightarrow (\varphi \rightarrow B^{\psi}[!\varphi]\chi)$$

is valid for beliefs after hard information.

Example 5.79 (Misleading with the truth) Consider a model where an agent believes that p, which is indeed true in the actual world to the far left, but for 'the wrong reason', viz. she thinks the most plausible world is the one to the far right. For convenience, assume that the final world also verifies a unique proposition letter q. The dashed links are knowledge links, the solid arrows are plausibility arrows, for the same agent.



Now giving the true information that we are not in the final world ($\neg q$) updates to:



in which the agent believes mistakenly that $\neg p$.

5.11 Outlook – Social Knowledge

Example 5.80 Imagine two generals who are planning a coordinated attack on a city. The generals are on two hills on opposite sides of the city, each with their armies, and they know they can only succeed in capturing the city if the two armies attack at the same time. But the valley that separates the two hills is in enemy hands, and any messenger that is sent from one army base to the other runs a severe risk of getting captured. The generals have agreed on a joint attack, but they still have to settle the time.

So the generals start sending messengers. General 1 sends a soldier with the message "We will attack tomorrow at dawn". Call this message p. Suppose his messenger gets across to general 2 at the other side of the valley. Then $\Box_2 p$ holds, but general 1 does not know this because he is uncertain about the transfer of his message. Now general 2 sends a messenger back to assure 1 that he has received his message. Suppose this messenger also gets across without being captured, then $\Box_1\Box_2 p$ holds. But general 2 does not know this, for he is uncertain about the success of transfer: $\neg\Box_2\Box_1\Box_2 p$. General 1 now sends a second messenger. If this one also safely delivers his message we have $\Box_2\Box_1\Box_2 p$. But general 1 does not know this, and so on, and so on. In this way, they'll continue sending messages infinitely (and certainly not attack tomorrow at dawn).

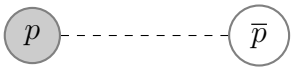
Clearly, this procedure will never establish common knowledge between the two generals. They share the knowledge of p but that is surely not enough for them to be convinced that

they will both attack at dawn. In case of real common knowledge every formula of the infinite set

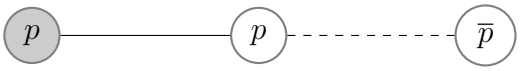
$$\{\Box_1 p, \Box_2 p, \Box_1\Box_2 p, \Box_2\Box_1 p, \Box_1\Box_2\Box_1 p, \Box_2\Box_1\Box_2 p, \ldots\}$$

holds.

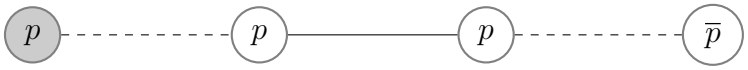
Here are pictures of how the situation as given in the previous example develops after each messenger delivers his message. Initially, general 1 settles the time of the attack. He knows that p but he also knows that general 2 does not know (with a dashed link for the accessibility of general 2):



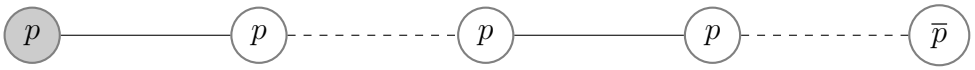
After the first messenger from 1 to 2 gets safely across we have (with a solid link for the accessibility relation of general 1):



After the message of 2 to 1 is safely delivered we get:



Successful transfer of the second message from 1 to 2 results in:



Note that in the second world from the left it does not hold that $\Box_2\Box_1\Box_2 p$, and therefore $\neg\Box_1\Box_2\Box_1\Box_2 p$ is true in the actual world.

The example makes it seem that achieving common knowledge is an extremely complicated or even impossible task. This conclusion is too negative, for common knowledge can be established immediately by public announcement. Suppose the two generals take a risk and get together for a meeting. Then general 1 simply says to general 2 "We will attack tomorrow at dawn", and immediately we get:

Still, we cannot express common knowledge between 1 and 2 by means of a single formula of our language. What we want to say is that the stacking of knowledge operators goes on indefinitely, but we have no formula for this.

The way to handle this is by adding a modality of common knowledge. $C_G\varphi$ expresses that it is common knowledge among the members of group G that φ. Here is the truth definition for it:

$$M, s \models C_G\varphi \quad \text{iff} \quad \begin{array}{l}\text{for all } t \text{ that are reachable from } s \\ \text{by some finite sequence of } \rightarrow_i \text{ steps } (i \in G)\text{: } M, t \models \varphi.\end{array}$$

Theorem 5.81 The complete epistemic logic with common knowledge is axiomatized by adding two axioms and a rule to the minimal epistemic logic. In the two axioms, E_G is used as an abbreviation for everybody in the group knows (defined as $E_G\varphi \leftrightarrow \Box_{g_1}\varphi \wedge \cdots \wedge \Box_{g_n}\varphi$, for all $g_1, \ldots, g_n$ in G):

Fixed-Point Axiom $C_G\varphi \leftrightarrow (\varphi \wedge E_G C_G\varphi)$.

Induction Axiom $(\varphi \wedge C_G(\varphi \rightarrow E_G\varphi)) \rightarrow C_G\varphi$.

C Necessitation Rule If φ is a theorem, then $C_G\varphi$ is also a theorem.

The axioms are also of independent interest for what they say. The Fixed-Point Axiom expresses an intuition of reflective equilibrium: common knowledge of φ is a proposition X implying φ of which every group member knows that X is true. On top of this, the Induction Axiom says that it is not just any equilibrium state of this kind, but the largest one.

To axiomatize PAL with common knowledge we need more expressive power. One possible (and elegant) way to achieve this is by adding an operator for *conditional common knowledge*, $C_G^\varphi\psi$, with the following truth definition:

$$M, s \models C_G^\varphi\psi \quad \text{iff} \quad \begin{array}{l}\text{for all } t \text{ that are reachable from } s \\ \text{by some finite sequence of } \rightarrow_i \text{ steps } (i \in G) \\ \text{through a series of states that all satisfy } \varphi \\ \text{it holds that } M, t \models \psi.\end{array}$$

This allows for a complete axiomatisation (again, we state the theorem without proof):

Theorem 5.82 PAL with conditional common knowledge is axiomatized completely by adding the valid reduction law

$$[!\varphi]C_G^\psi\chi \leftrightarrow (\varphi \rightarrow C_G^{\varphi \wedge [!\varphi]\psi}[!\varphi]\chi).$$

Example 5.83 Many social rituals are designed to create common knowledge. A prime example is cash withdrawal from a bank. You withdraw a large amount of money from your bank account and have it paid out to you in cash by the cashier. Typically, what happens is this. The cashier looks at you earnestly to make sure she has your full attention, and then she slowly counts out the banknotes for you: one thousand (counting ten notes while saying *one*, *two*, *three*, ..., *ten*), two thousand (counting another ten notes), three thousand (ten notes again), and four thousand (another ten notes). This ritual creates common knowledge that forty banknotes of 100 euros were paid out to you.

To see that this is different from mere knowledge, consider the alternative where the cashier counts out the money out of sight, puts it in an envelope, and hands it over to you. At home you open the envelope and count the money. Then the cashier and you have knowledge about the amount of money that is in the envelope. But the amount of money is not common knowledge among you. In order to create common knowledge you will have to insist on counting the money while the cashier is looking on, making sure that you have her full attention. For suppose you fail to do that. On recounting the money at home you discover there has been a mistake. One banknote is missing. Then the situation is as follows: the cashier believed that she knew there were forty banknotes. You now know there are only thirty-nine. How are you going to convince your bank that a mistake has been made, and that it is their mistake?

5.12 Outlook – Secrecy and Security

In computer science, protocols are designed and studied that do not reveal secret information to eavesdroppers. A strong property of such protocols is the following:

> Even if all communication is overheard, the secret is not compromised.

One example of how this can be achieved is given by the so-called *Dining Cryptographers Protocol*, designed by computer scientist David Chaum. The setting of this protocol is a situation where three cryptographers are eating out. At the end of the dinner, they are informed that the bill has been paid, either by one of them, or by NSA (the National Security Agency). Respecting each others' rights to privacy, they want to find out whether NSA paid or not, in such a way that in case one of them has paid the bill, the identity of the one who paid is not revealed to the two others.

They decide on the following protocol. Each cryptographer tosses a coin with his right-hand neighbour, with the result of the toss remaining hidden from the third person. Each cryptographer then has a choice between two public announcements: that the coins that she has observed agree or that they disagree. If she has not paid the bill she will say that they agree if the coins are the same and that they disagree otherwise; if she has paid the bill she will say the opposite: she will say that they agree if in fact they are different and she will say that they disagree if in fact they are the same. If everyone is speaking the

truth, the number of 'disagree' announcements will be even. This reveals that NSA has picked up the bill. If one person is 'lying', the number of 'disagree' announcements will be odd, indicating that one among them has paid the bill.

One can analyse this with epistemic logic by starting out with a model where the diners have common knowledge of the fact that either NSA or one of them has paid. Next, one updates with the result of the coin tosses, and with communicative acts representing the sharing of information between a cryptographer and his neighbour about these results.

Calling the cryptographers 1, 2 and 3, use p_1, p_2 and p_3 to express that 1, 2 or 3 has paid. The aim of the protocol is that everybody learns whether the formula $p_1 \vee p_2 \vee p_3$ is true or not, but if the formula is true, nobody (except the payer herself) should learn which of the three propositions was true. It is left to you to figure out why the above protocol achieves this goal.

Summary of Things You Have Learnt in This Chapter *You have learnt to look at information as uncertainty between various possible states of affairs, for cases of a single agent, but also for multi-agent settings that involve knowledge about knowledge. You know what information models are, and you are able to evaluate formulas from epistemic logic in information models. You have some experience with constructing formal proofs in epistemic logic. You are familiar with the concept of information update, and you can understand simple protocols designed to update information states. You have grasped the distinction between individual knowledge and common knowledge, and know in which cases public announcements can be used to establish common knowledge.*

Further Reading A classic on the logic of knowledge and belief is Jaakko Hintikka's [Hin62]. Epistemic logic for computer science is the subject of [MvdH95] and [FHMV95]. A textbook treatment of dynamic epistemic logic can be found in [DvdHK06]. A recent book on information exchange and interaction is [vB11].

Chapter 6

Logic and Action

Overview An action is something that takes place in the world, and that makes a difference to what the world looks like. Thus, actions are maps from states of the world to new states of the world. Actions can be of various kinds. The action of spilling coffee changes the state of your trousers. The action of telling a lie to your friend changes your friend's state of mind (and maybe the state of your soul). The action of multiplying two numbers changes the state of certain registers in your computer. Despite the differences between these various kinds of actions, we will see that they can all be covered under the same logical umbrella.

6.1 Actions in General

> Sitting quietly, doing nothing,
> Spring comes, and the grass grows by itself.
>
> *From: Zenrin kushu, compiled by Eicho (1429-1504)*

Action is change in the world. Change can take place by itself (see the poem above), or it can involve an agent who causes the change. You are an agent. Suppose you have a bad habit and you want to give it up. Then typically, you will go through various stages. At some point there is the *action* stage: you do what you have to do to effect a *change*.

Following instructions for how to combine certain elementary culinary actions (chopping an onion, firing up a stove, stirring the contents of a saucer) may make you a successful cook. Following instructions for how to combine communication steps may make you a successful salesperson, or a successful barrister. Learning to combine elementary computational actions in clever ways may make you a successful computer programmer.

Actions can often be characterized in terms of their results: "stir in heated butter and sauté until soft", "rinse until water is clear". In this chapter you will learn how to use logic for

analyzing the interplay of action and static descriptions of the world before and after the action.

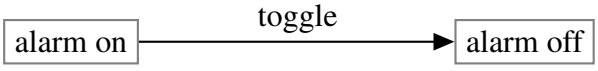
It turns out that *structured actions* can be viewed as compositions of basic actions, with only a few basic composition recipes: *conditional execution*, *choice*, *sequence*, and *repetition*. In some cases it is also possible to *undo* or reverse an action. This gives a further recipe: if you are editing a file, you can undo the last 'delete word' action, but you cannot undo the printing of your file.

Conditional or guarded execution ("remove from fire when cheese starts to melt"), sequence ("pour eggs in and swirl; cook for about three minutes; gently slide out of the pan"), and repetition ("keep stirring until soft") are ways in which a cook combines his basic actions in preparing a meal. But these are also the strategies for a lawyer when planning her defence ("only discuss the character of the defendant if the prosecution forces us", "first convince the jury of the soundness of the alibi, next cast doubt on the reliability of the witness for the prosecution"), or the basic layout strategies for a programmer in designing his code. In this chapter we will look at the logic of these ways of combining actions.

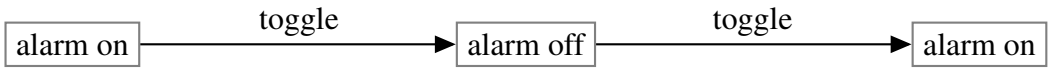
Action structure does not depend on the nature of the basic actions: it applies to *actions in the world*, such as preparing breakfast, cleaning dishes, or spilling coffee over your trousers. It also applies to *communicative actions*, such as reading an English sentence and updating one's state of knowledge accordingly, engaging in a conversation, sending an email with cc's, telling your partner a secret. These actions typically change the cognitive states of the agents involved. Finally, it applies to *computations*, i.e., actions performed by computers. Examples are computing the factorial function, computing square roots, etc. Such actions typically involve changing the memory state of a machine. Of course there are connections between these categories. A communicative action will usually involve some computation involving memory, and the utterance of an imperative ('Shut the door!') is a communicative action that is directed towards action in the world.

There is a very general way to model action and change, a way that we have in fact seen already. The key is to view a changing world as a set of situations linked by labeled arcs. In the context of epistemic logic we have looked at a special case of this, the case where the arcs are epistemic accessibility relations: agent relations that are reflexive, symmetric, and transitive. Here we drop this restriction.

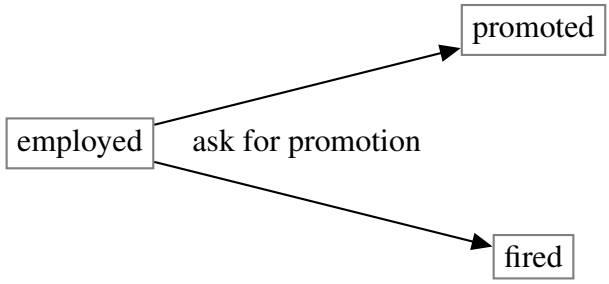
Consider an action that can be performed in only one possible way. Toggling a switch for switching off your alarm clock is an example. This can be pictured as a transition from an initial situation to a new situation:



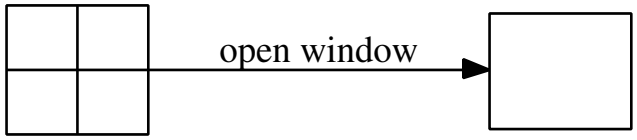
Toggling the switch once more will put the alarm back on:



Some actions do not have determinate effects. Asking your boss for a promotion may get you promoted, but it may also get you fired, so this action can be pictured like this:

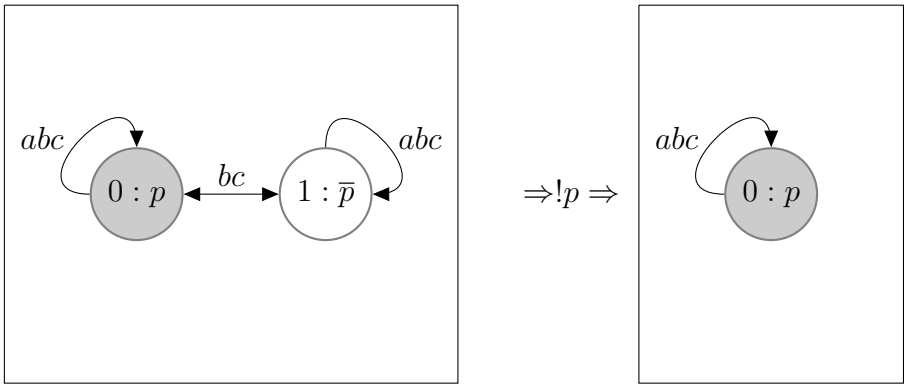


Another example: opening a window. This brings about a change in the world, as follows.



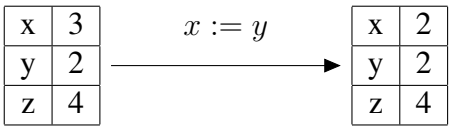
The action of window-opening changes a state in which the window is closed into one in which it is open. This is more subtle than toggling an alarm clock, for once the window is open a *different* action is needed to close it again. Also, the action of opening a window can only be applied to *closed* windows, not to open ones. We say: performing the action has a *precondition* or *presupposition*.

In fact, the public announcements from the previous chapter can also be viewed as (communicative) actions covered by our general framework. A public announcement is an action that effects a change in an information model.



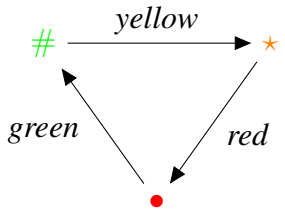
On the left is an epistemic situation where p is in fact the case (indicated by the grey shading), but b and c cannot distinguish between the two states of affairs, for they do not know whether p. If in such a situation there is a public announcement that p is the case, then the epistemic situation changes to what is pictured on the right. In the new situation, everyone knows that p is the case, and everyone knows that everyone knows, and so on. In other words: p has become common knowledge.

Here is computational example. The situation on the left in the picture below gives a highly abstract view of part of the memory of a computer, with the contents of three registers x, y and z. The effect of the assignment action $x := y$ on this situation is that the old contents of register x gets replaced by the contents of register y. The result of the action is the picture on the right.



The command to put the value of register y in register x makes the contents of registers x and y equal.

The next example models a traffic light that can turn from green to yellow to red and again to green. The transitions indicate which light is turned on (the light that is currently on is switched off). The state # is the state with the green light on, the state ⋆ the state with the yellow light on, and the state • the state with the red light on.



These examples illustrate that it is possible to approach a wide variety of kinds of actions from a unified perspective. In this chapter we will show that this is not only possible, but also fruitful.

In fact, much of the reasoning we do in everyday life is reasoning about change. If you reflect on an everyday life problem, one of the things you can do is run through various scenarios in your mind, and see how you would (re)act if things turn out as you imagine. Amusing samples are in the Dutch 'Handboek voor de Moderne Vrouw' (The Modern Woman's Handbook). See `http://www.handboekvoordemodernevrouw.nl/`.

Here is a sample question from 'Handboek voor de Moderne Vrouw': 'I am longing for a cosy Xmas party. What can I do to make our Xmas event happy and joyful?' Here is the recommendation for how to reflect on this:

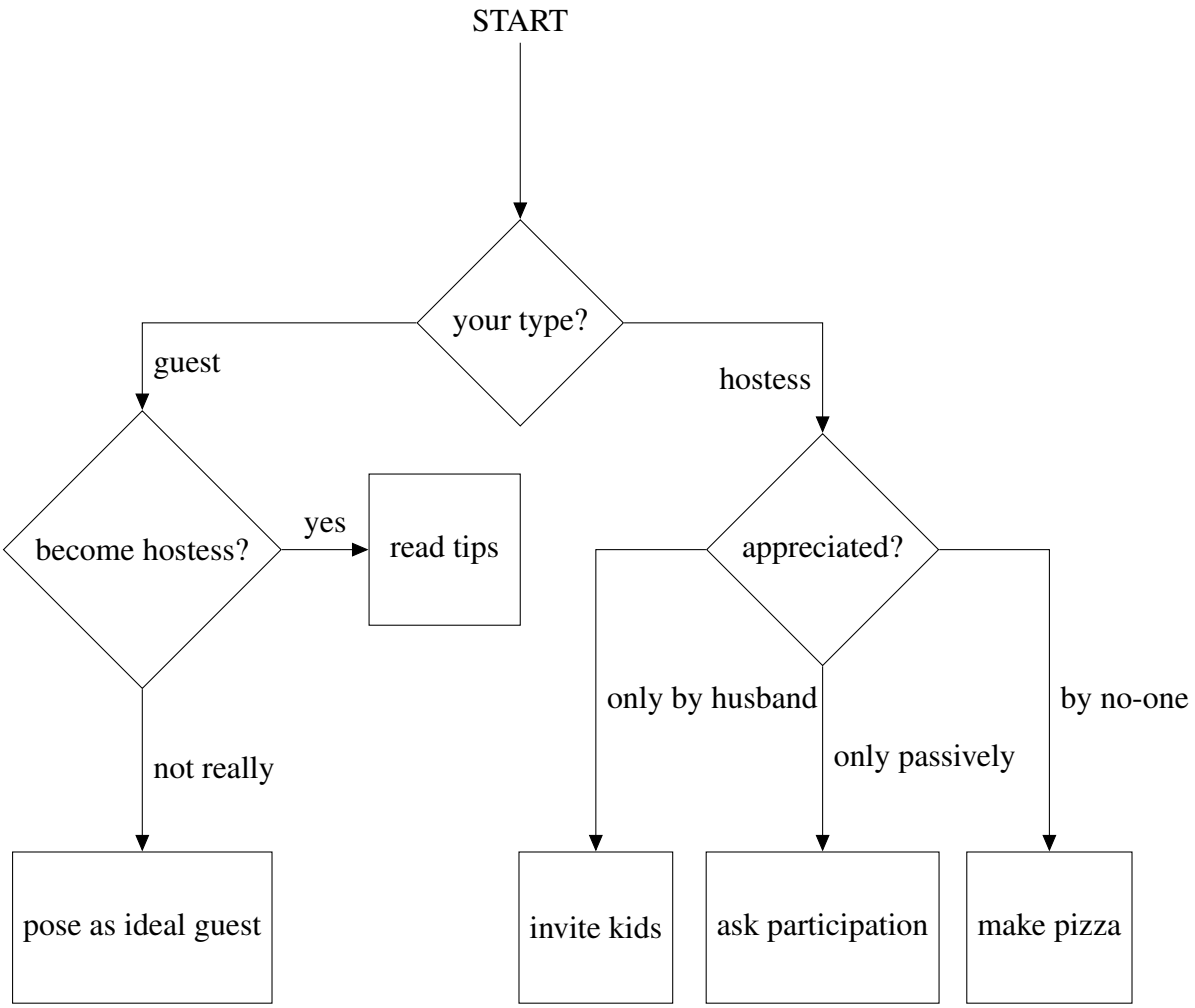


Figure 6.1: Flow Diagram of ‘Happy Xmas Procedure’

Are you the type of a 'guest' or the type of a 'hostess'?
  If the answer is 'guest': Would you like to become a hostess?
    If the answer is 'not really' then
      your best option is to profile as an ideal guest
      and hope for a Xmas party invitation elsewhere.
    If the answer is 'yes' then here are some tips on how to become a great hostess: ...
  If the answer is 'hostess', then ask yourself:
    Are your efforts truly appreciated?
      If the answer is 'Yes, but only by my own husband' then
        probably your kids are bored to death.
        Invite friends with kids of the same age as yours.
      If the answer is 'Yes, but nobody lifts a finger to help out' then
        Ask everyone to prepare one of the courses.
      If the answer is 'No, I only gets moans and sighs' then
        put a pizza in the microwave for your spouse and kids
        and get yourself invited by friends.

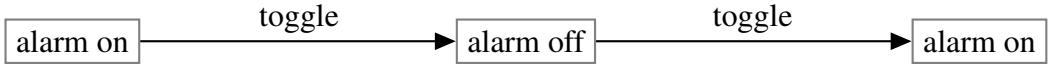
Figure 6.1 gives a so-called flow diagram for the recommendations from this example. Note that the questions are put in $\diamond$ boxes, that the answers are labels of outgoing arrows of the $\diamond$ boxes, and that the actions are put in $\square$ boxes.

6.2 Sequence, Choice, Repetition, Test

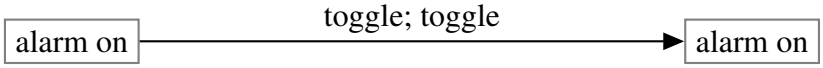
In the logic of propositions, the natural operations are *not*, *and* and *or*. These operations are used to map truth values into other truth values. When we want to talk about action, the repertoire of operations gets extended. What are natural things to do with actions?

When we want to talk about action at a very general level, then we first have to look at how actions can be structured. Let's assume that we have a set of basic actions. Call these basic actions a, b, c, and so on. Right now we are not interested in the internal structure of basic actions. The actions a, b, c could be anything: actions in the world, basic acts of communication, or basic changes in the memory state of a computer. Given such a set of basic actions, we can look at natural ways to combine them.

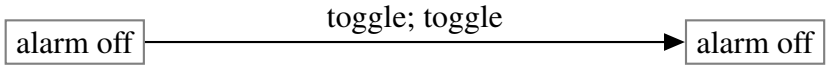
Sequence In the first place we can *perform one action after another*: first eat breakfast, then do the dishes. First execute action a, next execute action b. First toggle a switch. Then toggle it again. Consider again the alarm clock toggle action.



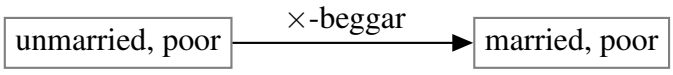
Writing the sequence of two actions a and b as $a; b$, we get:

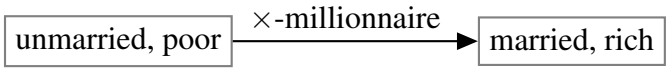


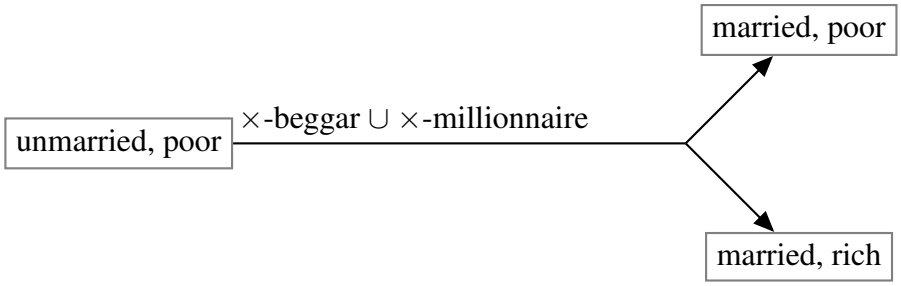
Starting out from the situation where the alarm is off, we would get:



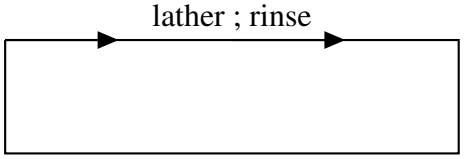
Choice A complex action can also consist of a *choice* between simpler actions: either drink tea or drink coffee. Either marry a beggar or marry a millionnaire.



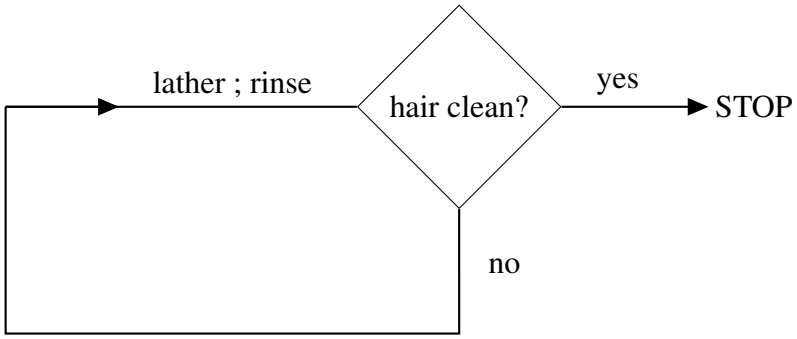




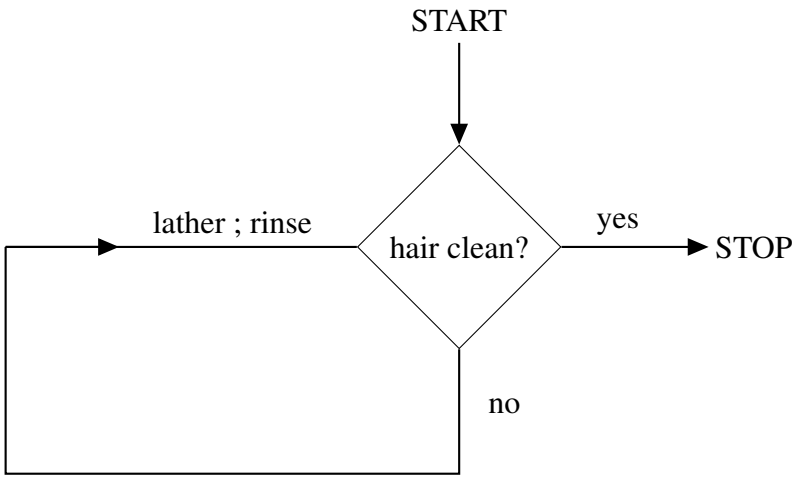
Repetition Actions can be repeated. The phrase 'lather, rinse, repeat' is used as a joke at people who take instructions too literally: the stop condition 'until hair is clean' is omitted. There is also a joke about an advertising executive who increases the sales of his client's shampoo by adding the word 'repeat' to its instructions. If taken literally, the compound action 'lather, rinse, repeat' would look like this:



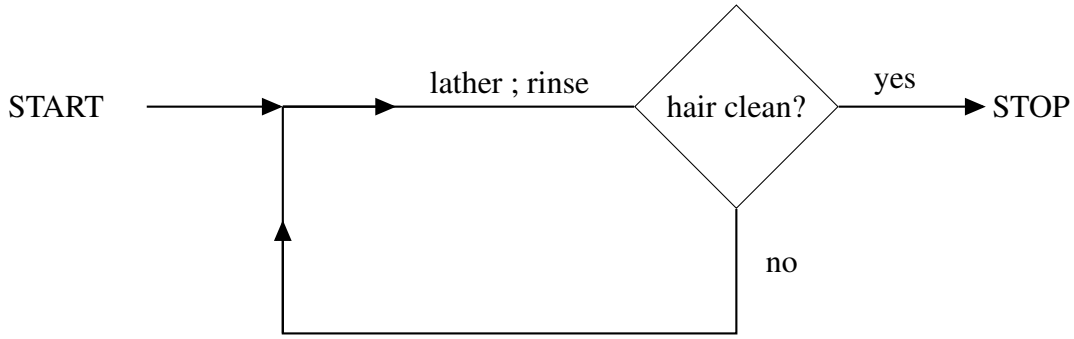
Repeated actions usually have a *stop condition*: repeat the lather rinse sequence until your hair is clean. This gives a more sensible interpretation of the repetition instruction:



Looking at the picture, we see that this procedure is ambiguous, for where do we start? Here is one possibility:



And here is another:



The difference between these two procedures is that the first one starts with a ‘hair clean?’ check: if the answer is ‘yes’, nothing happens. The second procedure starts with a ‘lather; rinse’ sequence, no matter the initial state of your hair.

In many programming languages, this same distinction is made by means of a choice between two different constructs for expressing 'condition controlled loops':

while `not hair_clean` **do** { `lather; rinse` }

repeat { `lather ; rinse` } **until** `hair_clean`

The first loop does not guarantee that the 'lather ; rinse' sequence gets performed at least once; the second loop does.

Test The 'condition' in a condition-controlled loop (the condition 'hair_clean', for example) can itself be viewed as an action: a test whether a certain fact holds. A test to see whether some condition holds can also be viewed as a basic action. Notation for the action that tests condition φ is $?\varphi$. The question mark turns a formula (something that can be true or false) into an action (something that can succeed or fail).

If we express tests as $?\varphi$, then we should specify the language from which φ is taken. Depending on the context, this could be the language of propositional logic, the language of predicate logic, the language of epistemic logic, and so on.

Since we are taking an abstract view, the basic actions can be anything. Still, there are a few cases of basic action that are special. The action that always succeeds is called *SKIP*. The action that always fails is called *ABORT*. If we have tests, then clearly *SKIP* can be expressed as $?\top$ (the test that always succeeds) and *ABORT* as $?\bot$ (the test that always fails).

Using test, sequence and choice we can express the familiar 'if then else' from many programming languages.

if `hair_clean` **then** `skip` **else** { `lather ; rinse` }

This becomes a choice between a test for clean hair (if this test succeeds then nothing happens) and a sequence consisting of a test for not-clean-hair followed by a lather and a rinse (if the hair is not clean then it is first lathered and then rinsed).

`?hair_clean` $\cup$ { `?¬hair_clean ; lather ; rinse` }

The general recipe for expressing **if** φ **then** α_1 **else** α_2 is given by:

$$?\varphi; \alpha_1 \ \cup \ ?\neg\varphi; \alpha_2.$$

Since exactly one of the two tests $?\varphi$ and $?\neg\varphi$ will succeed, exactly one of α_1 or α_2 will get executed.

Using the operation for turning a formula into a test, we can first test for p and next test for q by means of $?p; ?q$. Clearly, the order of testing does not matter, so this is equivalent to $?q; ?p$. And since the tests do not change the current state, this can also be expressed as a single test $?(p \wedge q)$.

Similarly, the choice between two tests $?p$ and $?q$ can be written as $?p \cup ?q$. Again, this is equivalent to $?q \cup ?p$, and it can be turned into a single test $?(p \vee q)$.

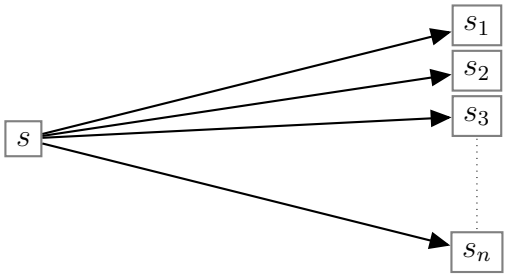
Converse Some actions can be undone by reversing them: the reverse of opening a window is closing it. Other actions are much harder to undo: if you smash a piece of china then it is sometimes hard to mend it again. So here we have a choice: do we assume that basic actions can be undone? If we do, we need an operation for this, for taking the *converse* of an action. If, in some context, we assume that undoing an action is generally impossible we should omit the *converse* operation in that context.

Exercise 6.1 Suppose $\breve{}$ is used for reversing basic actions. So $a\breve{}$ is the converse of action a, and $b\breve{}$ is the converse of action b. Let $a; b$ be the sequential composition of a and b, i.e., the action that consists of first doing a and then doing b. What is the converse of $a; b$?

6.3 Viewing Actions as Relations

As an exercise in abstraction, we will now view actions as binary relations on a set S of states. The intuition behind this is as follows. Suppose we are in some state s in S. Then performing some action a will result in a new state that is a member of some set of new states $\{s_1, \ldots, s_n\}$.

If this set is empty, this means that the action a has aborted in state s. If the set has a single element s', this means that the action a is deterministic on state s, and if the set has two or more elements, this means that action a is non-deterministic on state s. The general picture is:



Clearly, when we extend this picture to the whole set S, what emerges is a binary relation on S, with an arrow from s to s' (or equivalently, a pair (s, s') in the relation) just in case performing action a in state s may have s' as result. Thus, we can view binary relations on S as the interpretations of basic action symbols a.

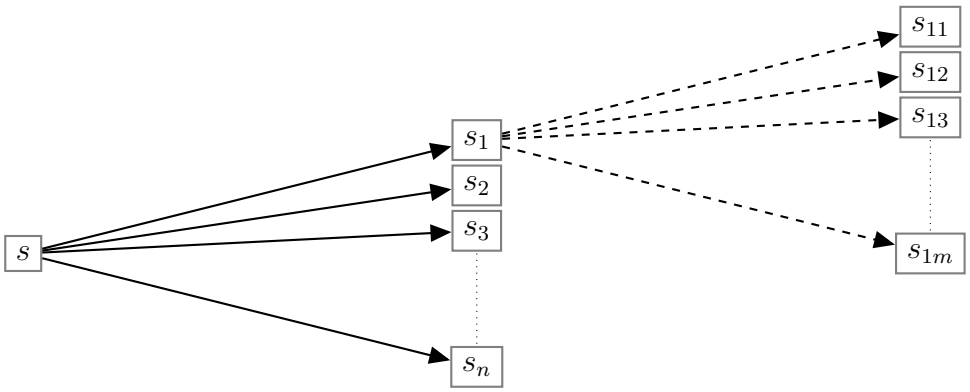
The set of all pairs taken from S is called $S \times S$, or S^2. A binary relation on S is simply a set of pairs taken from S, i.e., a subset of S^2.

Given this abstract interpretation of basic relations, it makes sense to ask what corresponds to the operations on actions that we encountered in Section 6.2. Let's consider them in turn.

Sequence Given that action symbol a is interpreted as binary relation R_a on S, and that action symbol b is interpreted as binary relation R_b on S, what should be the interpretation of the action sequence $a; b$? Intuitively, one can move from state s to state s' just in case there is some intermediate state s_0 with the property that a gets you from s to s_0 and b gets you from s_0 to s'. This is a well-known operation on binary relations, called *relational composition*. If R_a and R_b are binary relations on the same set S, then $R_a \circ R_b$ is the binary relation on S given by:

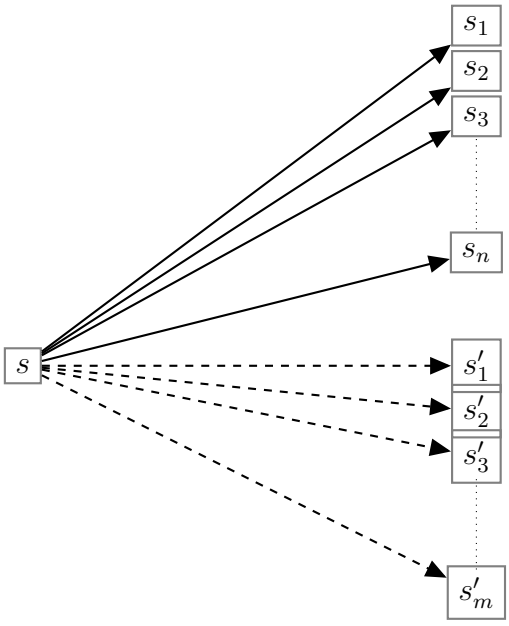
$$R_a \circ R_b = \{(s, s') \mid \text{ there is some } s_0 \in S : (s, s_0) \in R_a \text{and } (s_0, s') \in R_b\}.$$

If basic action symbol a is interpreted as relation R_a, and basic action symbol b is interpreted as relation R_b, then the sequence action $a; b$ is interpreted as $R_a \circ R_b$. Here is a picture:



If the solid arrows interpret action symbol a and the dashed arrows interpret action symbol b, then the arrows consisting of a solid part followed by a dashed part interpret the sequence $a; b$.

Choice Now suppose again that we are in state s, and that performing action a will get us in one of the states in $\{s_1, \ldots, s_n\}$. And supposse that in that same state s, performing action b will get us in one of the states in $\{s'_1, \ldots, s'_m\}$.



Then performing action $a \cup b$ (the choice between a and b) in s will get you in one of the states in $\{s_1, \ldots, s_n\} \cup \{s'_1, \ldots, s'_m\}$. More generally, if action symbol a is interpreted as the relation R_a, and action symbol b is interpreted as the relation R_b, then $a \cup b$ will be interpreted as the relation $R_a \cup R_b$ (the union of the two relations).

Test A notation that is often used for the equality relation (or: identity relation is I. The binary relation I on S is by definition the set of pairs given by:

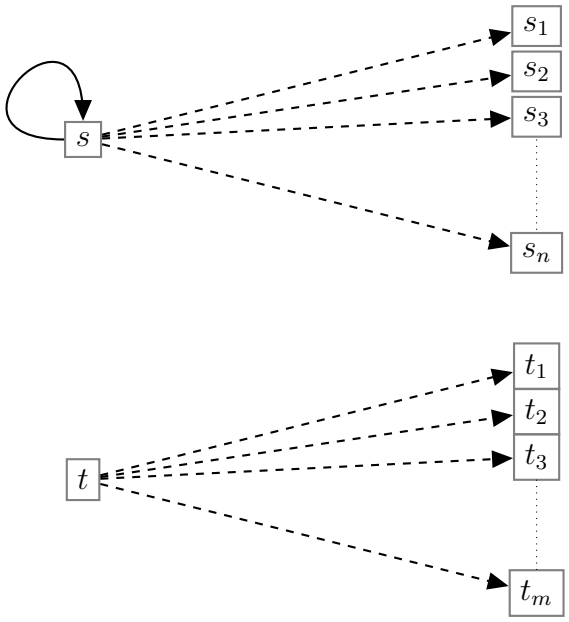
$$I = \{(s, s) \mid s \in S\}.$$

A test $?\varphi$ is interpreted as a subset of the identity relation, namely as the following set of pairs:

$$R_{?\varphi} = \{(s, s) \mid s \in S, s \models \varphi\}$$

From this we can see that a test does not change the state, but checks whether the state satisfies a condition.

To see the result of combining a test with another action:



The solid arrow interprets a test $?\varphi$ that succeeds in state s but fails in state t. If the dashed arrows interpret a basic action symbol a, then, for instance, (s, s_1) will be in the interpretation of $?\varphi; a$, but (t, t_1) will not.

Since $\top$ is true in any situation, we have that $?\top$ will get interpreted as I (the identity relation on S). Therefore, $?\top; a$ will always receive the same interpretation as a.

Since $\bot$ is false in any situation, we have that $?\bot$ will get interpreted as $\emptyset$ (the empty relation on S). Therefore, $?\bot; a$ will always receive the same interpretation as $?\bot$.

Before we handle repetition, it is useful to switch to a more gereral perspective.

6.4 Operations on Relations

Relations were introduced in Chapter 4 on predicate logic. In this chapter we view actions as binary relations on a set S of situations. Such a binary relation is a subset of $S \times S$, the set of all pairs (s, t) with s and t taken from S. It makes sense to develop the general topic of operations on binary relations. Which operations suggest themselves, and what are the corresponding operations on actions?

In the first place, there are the usual set-theoretic operations. Binary relations are sets of pairs, so taking unions, intersections and complements makes sense (also see Appendix A). We have already seen that taking unions corresponds to choice between actions.

Example 6.2 The union of the relations 'mother' and 'father' is the relation 'parent'.

Example 6.3 The intersection of the relations $\subseteq$ and $\supseteq$ is the equality relation $=$.

In Section 6.3 we encountered the notation I for the equality (or: identity) relation on a set S. We have seen that tests get interpreted as subsets of I.

We also looked at composition of relations. $R_1 \circ R_2$ is the relation that performing an R_1 step followed by an R_2 step. To see that order of composition matters, consider the following example.

Example 6.4 The relational composition of the relations 'mother' and 'parent' is the relation 'grandmother', for 'x is grandmother of y' means that there is a z such that x is mother of z, and z is parent of y.

The relational composition of the relations 'parent' and 'mother' is the relation 'maternal grandparent', for 'x is maternal grandparent of y' means that there is a z such that x is parent of z and z is mother of y.

Exercise 6.5 What is the relational composition of the relations 'father' and 'mother'?

Another important operation is *relational converse*. The relational converse of a binary relation R, notation $R^\smile$, is the relation given by:

$$R^\smile = \{(y, x) \in S^2 \mid (x, y) \in R\}.$$

Example 6.6 The relational converse of the 'parent' relation is the 'child' relation.

Exercise 6.7 What is the relational converse of the $\subseteq$ relation?

The following law describes the interplay between composition and converse:

Converse of composition $(R_1 \circ R_2)^\smile = {R_2}^\smile \circ {R_1}^\smile$.

Exercise 6.8 Check from the definitions that $(R_1 \circ R_2)^\smile = {R_2}^\smile \circ {R_1}^\smile$ is valid.

There exists a long list of logical principles that hold for binary relations. To start with, there are the usual Boolean principles that hold for all sets:

Commutativity $R_1 \cup R_2 = R_2 \cup R_1$, $R_1 \cap R_2 = R_2 \cap R_1$,

Idempotence $R \cup R = R$, $R \cap R = R$.

Laws of De Morgan $\overline{R_1 \cup R_2} = \overline{R_1} \cap \overline{R_2}$, $\overline{R_1 \cap R_2} = \overline{R_1} \cup \overline{R_2}$.

Specifically for relational composition we have:

Associativity $R_1 \circ (R_2 \circ R_3) = (R_1 \circ R_2) \circ R_3$.

Distributivity

$$\begin{aligned} R_1 \circ (R_2 \cup R3) &= (R_1 \circ R_2) \cup (R_1 \circ R_3) \\ (R_1 \cup R_2) \circ R3) &= (R_1 \circ R_3) \cup (R_2 \circ R_3). \end{aligned}$$

There are also many principles that seem plausible but that are invalid. To see that a putative principle is invalid one should look for a counterexample.

Example 6.9 $R \circ R = R$ is invalid, for if R is the 'parent' relation, then the principle would state that 'grandparent' equals 'parent', which is false.

Exercise 6.10 Show by means of a counterexample that $R_1 \cup (R_2 \circ R_3) = (R_1 \cup R_2) \circ (R_1 \cup R_3)$ is invalid.

Exercise 6.11 Check from the definitions that $R_1 \circ (R_2 \cup R_3) = (R_1 \circ R_2) \cup (R_1 \circ R_3)$ is valid.

Exercise 6.12 Check from the definition that $R^{\smile} = R$ is valid.

Exercise 6.13 Check from the definitions that $(R_1 \cup R_2)^{\smile} = {R_1}^{\smile} \cup {R_2}^{\smile}$ is valid.

Transitive Closure A relation R is transitive if it holds that if you can get from x to y in two R-steps, then it is also possible to get from x to y in a single R-step (see page 4-20 above). This can be readily expressed in terms of relational composition.

R is transitive iff $R \circ R \subseteq R$.

The *transitive closure* of a relation R is defined as the smallest transitive relation S that contains R. This means: S is the transitive closure of R if

(1) $R \subseteq S$,

(2) $S \circ S \subseteq S$,

(3) if $R \subseteq T$ and $T \circ T \subseteq T$ then $S \subseteq T$.

Requirement (1) expresses that R is contained in S, requirement (2) expresses that S is transitive, and requirement (3) expresses that S is the *smallest* transitive relation that contains R: any T that satisfies the same requirements must be at least as large as S.

The customary notation for the transitive closure of R is R^+. Here is an example.

Example 6.14 The transitive closure of the 'parent' relation is the 'ancestor' relation. If x is parent of y then x is ancestor of y, so the parent relation is contained in the ancestor relation. If x is an ancestor of y and y is an ancestor of z then surely x is an ancestor of z, so the ancestor relation is transitive. Finally, the ancestor relation is the smallest transitive relation that contains the parent relation.

You can think of a binary relation R as a recipe for taking R-steps. The recipe for taking double R-steps is now given by $R \circ R$. The recipe for taking triple R-steps is given by $R \circ R \circ R$, and so on.

There is a formal reason why the order of composition does not matter: $R_1 \circ (R_2 \circ R_3)$ denotes the same relation as $(R_1 \circ R_2) \circ R_3$. because of the above-mentioned principle of associativity.

The n-fold composition of a binary relation R on S with itself can be defined from R and I (the identity relation on S), by recursion (see Appendix, Section A.6), as follows:

$$\begin{aligned} R^0 &= I \\ R^n &= R \circ R^{n-1} \text{ for } n > 0. \end{aligned}$$

Abbreviation for the n-fold composition of R is R^n. This allows us to talk about taking a specific number of R-steps.

Notice that $R \circ I = R$. Thus, we get that $R^1 = R \circ R^0 = R \circ I = R$.

The transitive closure of a relation R can be computed by means of:

$$R^+ = R \cup R^2 \cup R^3 \cup \cdots$$

This can be expressed without the $\cdots$, as follows:

$$R^+ = \bigcup_{n \in \mathbb{N}, n>0} R^n.$$

Thus, R^+ denotes the relation of doing an arbitrary finite number of R-steps (at least one).

Closely related to the transitive closure of R is the reflexive transitive closure of R. This is, by definition, the smallest relation that contains R and that is both reflexive and transitive. The reflexive transitive closure of R can be computed by:

$$R^* = I \cup R \cup R^2 \cup R^3 \cup \cdots$$

This can be expressed without the $\cdots$, as follows:

$$R^* = \bigcup_{n \in \mathbb{N}} R^n.$$

Thus, R^* denotes the relation of doing an arbitrary finite number of R-steps, including zero steps.

Notice that the following holds:

$$R^+ = R \circ R^*.$$

Exercise 6.15 The following identity between relations is not valid:

$$(R \cup S)^* = R^* \circ S^*.$$

Explain why not by giving a counter-example.

Exercise 6.16 The following identity between relations is not valid:

$$(R \circ S)^* = R^* \circ S^*.$$

Explain why not by giving a counter-example.

For Loops In programming, repetition consisting of a specified number of steps is called a *for loop*. Here is an example of a loop for printing ten lines, in the programming language *Ruby*:

```
#!/usr/bin/ruby

for i in 0..10
   puts "Value of local variable is #{i}"
end
```

If you have a system with *Ruby* installed, you can save this as a file and execute it.

While Loops, Repeat Loops If R is the interpretation of a ('doing a once'), then R^* is the interpretation of 'doing a an arbitrary finite number of times', and R^+ is the interpretation of 'doing a an arbitrary finite number of times but at least once'. These relations can be used to define the interpretation of *while loops* and *repeat loops* (the so-called *condition controlled loops*), as follows.

If a is interpreted as R_a, then the condition-controlled loop 'while φ do a' is interpreted as:

$$(R_{?\varphi} \circ R_a)^* \circ R_{?\neg\varphi}.$$

First do a number of steps consisting of a $?\varphi$ test followed by an a action, next check that $\neg\varphi$ holds.

Exercise 6.17 Supposing that a gets interpreted as the relation R_a, $?\varphi$ as $R_{?\varphi}$ and $?\neg\varphi$ as $R_{?\neg\varphi}$, give a relational interpretation for the condition controlled loop 'repeat a until φ'.

6.5 Combining Propositional Logic and Actions: PDL

The language of propositional logic over some set of basic propositions P is given by:

$$\varphi \ ::= \ \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \text{ where } p \text{ ranges over } P.$$

If we assume that a set of basic action symbols A is given, then the language of actions that we discussed in Sections 6.2 and 6.3 above can be formally defined as:

$$\alpha \ ::= \ a \mid ?\varphi \mid \alpha; \alpha \mid \alpha \cup \alpha \mid \alpha^* \text{ where } a \text{ ranges over } A.$$

Note that the test $?\varphi$ in this definition refers to the definition of φ in the language of propositional logic. Thus, the language of propositional logic is embedded in the language of actions.

Now here is a new idea, for also doing the converse: extend the language of propositional logic with a construction that describes the results of executing an action α.

If α is interpreted as a binary relation then in a given state s there may be several states s' for which (s, s') is in the interpretation of α.

Interpret $\langle\alpha\rangle\varphi$ as follows:

> $\langle\alpha\rangle\varphi$ is true in a state s if for *some* s' with (s, s') in the interpretation of α it holds that φ is true in s'.

For instance, if a is the action of asking for promotion, and p is the proposition expressing that one is promoted, then $\langle a\rangle p$ expresses that asking for promotion may result in actually getting promoted.

Another useful expression is $[\alpha]\varphi$, with the following interpretation:

> $[\alpha]\varphi$ is true in a state s if for *every* s' with (s, s') in the interpretation of α it holds that φ is true in s'.

For instance, if a again expresses asking for promotion, and p expresses that one is promoted, then $[a]p$ expresses that, in the current state, the action of asking for a promotion *always* results in getting promoted.

Note that $\langle a\rangle p$ and $[a]p$ are not equivalent: think of a situation where asking for a promotion may also result in getting fired. In that case $\langle a\rangle p$ may still hold, but $[a]p$ does not hold.

If one combines propositional logic with actions in this way one gets a basic logic of change called Propositional Dynamic Logic or PDL. Here is the formal definition of the language of PDL:

Definition 6.18 (Language of PDL — propositional dynamic logic) Let p range over the set of basic propositions P, and let a range over a set of basic actions A. Then the formulas φ and action statements α of propositional dynamic logic are given by:

$$\begin{array}{lcl} \varphi & ::= & \top \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \varphi_1 \wedge \varphi_2 \mid \langle\alpha\rangle\varphi \mid [\alpha]\varphi \\ \alpha & ::= & a \mid ?\varphi \mid \alpha_1; \alpha_2 \mid \alpha_1 \cup \alpha_2 \mid \alpha^* \end{array}$$

The definition does not have $\rightarrow$ or $\leftrightarrow$. But this does not matter, for we can introduce these operators by means of abbreviations or shorthands.

$\top$ is the formula that is always true. From this, we can define $\bot$, as shorthand for $\neg\top$.

Similarly, $\varphi_1 \rightarrow \varphi_2$ is shorthand for $\neg\varphi_1 \vee \varphi_2$, $\varphi_1 \leftrightarrow \varphi_2$ is shorthand for $(\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)$.

Propositional dynamic logic abstracts over the set of basic actions, in the sense that basic actions can be anything. In the language of PDL they are atoms. This means that the range of applicability of PDL is vast. The only thing that matters about a basic action a is that it is interpreted by some binary relation on a state set.

Propositional dynamic logic has two basic syntactic categories: *formulas* and *action statements*. Formulas are used for talking about states, action statements are used for classifying transitions between states. The same distinction between formulas and action statements can be found in all imperative programming languages. The statements of C or Java or Ruby are the action statements. Basic actions in C are assigning a value to a variable. These are instructions to change the memory state of the machine. The so-called Boolean expressions in C behave like formulas of propositional logic. They appear as conditions or tests in conditional expressions. Consider the following C statement:

```
if (y < z)
  x = y;
else
  x = z;
```

This is a description of an action. But the ingredient `(y<z)` is not a statement (description of an action) but a Boolean expression (description of a state) that expresses a test.

Propositional dynamic logic is an extension of propositional logic with action statements, just like epistemic logic is an extension of propositional logic with epistemic modalities. Let a set of basic propositions P be given. Then appropriate states will contain valuations for these propositions. Let a set of basic actions A be given. Then every basic action corresponds to a binary relation on the state set. Together this gives a labeled transition system with valuations on states as subsets from P and labels on arcs between states taken from A.

Exercise 6.19 Suppose we also want to introduce a shorthand α^n, for a sequence of n copies of action statement α. Show how this can be defined by induction. (Hint: use $\alpha^0 := ?\top$ as the base case.)

Let's get a feel for the kind of things we can express with PDL. For any action statement α,

$$\langle\alpha\rangle\top$$

expresses that the action α has at least one successful execution. Similarly,

$$[\alpha]\bot$$

expresses that the action fails (cannot be executed in the current state).

The basic actions can be anything, so let us focus on a basic action a that is interpreted as the relation R_a. Suppose we want to say that some execution of a leads to a p state and another execution of a leads to a non-p state. Then here is a PDL formula for that:

$$\langle a\rangle p \wedge \langle a\rangle \neg p.$$

If this formula is true in a state s, then this means that R_a forks in that state: there are at least two R_a arrows starting from s, one of them to a state s_1 satisfying p and one of them to a state s_1 that does not satisfy p. For the interpretation of P we need properties of states, for p is like a one-place predicate in predicate logic.

If the basic actions are changes in the world, such as spilling milk S or cleaning C, then $[C;S]d$ expresses that cleaning up followed by spilling milk always results in a dirty state, while $[S;C]\neg d$ expresses that the occurrence of these events in the reverse order always results in a clean state.

6.6 Transition Systems

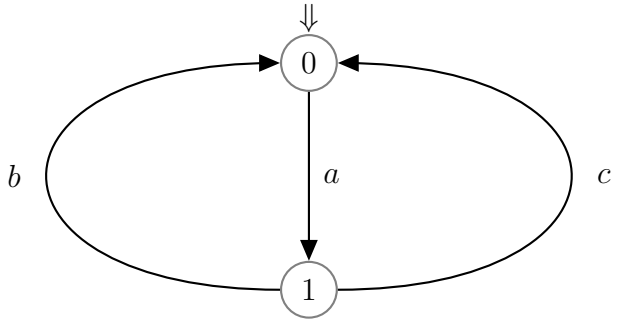
In Section 6.7 we will define the semantics of PDL relative to labelled transition systems, or process graphs.

Definition 6.20 (Labelled transition system) Let P be a set of basic propositions and A a set of labels for basic actions. Then a labelled transition system (or LTS) over atoms P and agents A is a triple $M = \langle S, R, V\rangle$ where S is a set of states, $V : S \to \mathcal{P}(P)$ is a valuation function, and $R = \{\stackrel{a}{\to} \subseteq S \times S \mid a \in A\}$ is a set of labelled transitions, i.e., a set of binary relations on S, one for each label a.

Another way to look at a labelled transition system is as a first order model predicate for a language with unary and binary predicates.

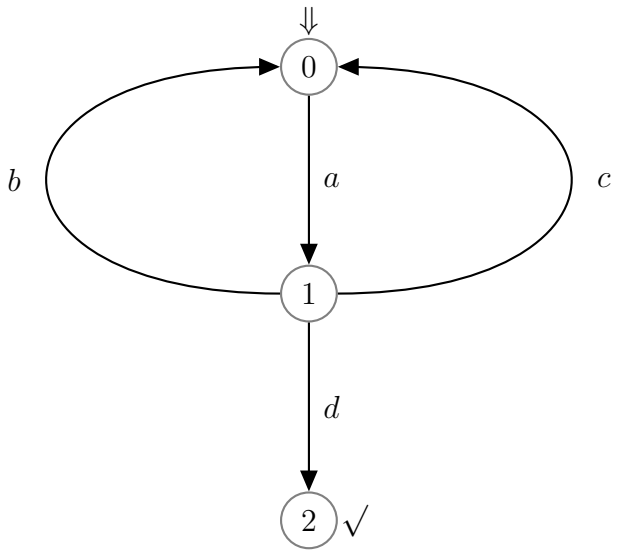
LTSs with a designated node (called the *root node*) are called *pointed LTSs* or *process graphs*.

The process of repeatedly doing a, followed by a choice between b and c can be viewed as a process graph, as follows:



The root note 0 is indicated by $\Downarrow$. There are two states 0 and 1. The process start in state 0 with the execution of action a. This gets us to state 1, where there are two possible actions b and c, both of which get us back to state 0, and there the process repeats. This is an infinite process, just like an operating system of a computer. Unless there is a system crash, the process goes on forever.

Jumping out of a process can be done by creating an action that moves to an end state.



We can think about $\surd$ as a proposition letter, and then use PDL to talk about these process graphs. In state 1 of the first model $\langle d\rangle\surd$ is false, in state 1 of the second model $\langle d\rangle\surd$ is true. This formula expresses that a d transition to a $\surd$ state is possible.

In both models it is the case in state 0 that after *any number* of sequences consisting of an a step followed by either a b or a c step, a further a step is possible. This is expressed by the following PDL formula: $[(a;(b\cup c))^*]\langle a\rangle\top$.

Exercise 6.21 Which of the following formulas are true in state 0 of the two models given above:

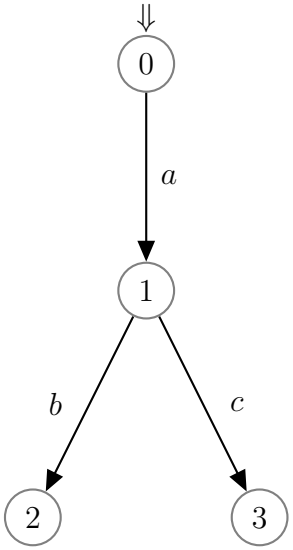
(1) $\langle a;d\rangle\surd$.

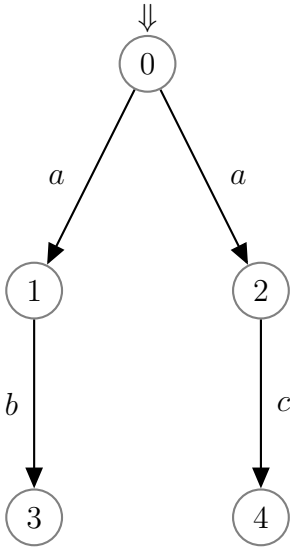
(2) $[a;d]\surd$.

(3) $[a](\langle b\rangle\top\wedge\langle c\rangle\top)$.

(4) $[a]\langle d\rangle\surd$.

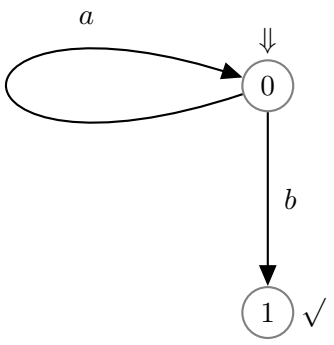
The following two pictures illustrate an important distinction:

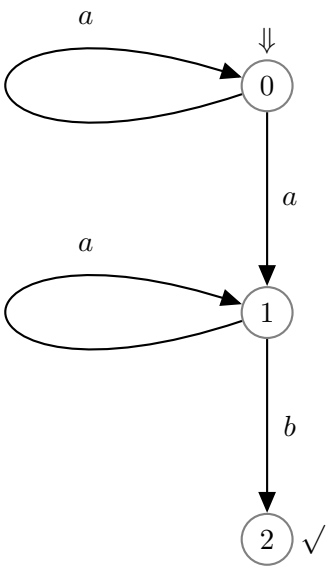




In the picture on the left, it is possible to take an a action from the root, and next to make a choice between doing b or doing c. In the picture on the right, there are two 'ways' of doing a, one of them ends in a state where b is the only possible move and the other one ending in a state where c is the only possible move. This difference can be expressed in a PDL formula, as follows. In the root state of the picture on the left, $[a](\langle b\rangle\top \wedge \langle c\rangle\top)$ is true, in the root state of the picture on the right this formula is false.

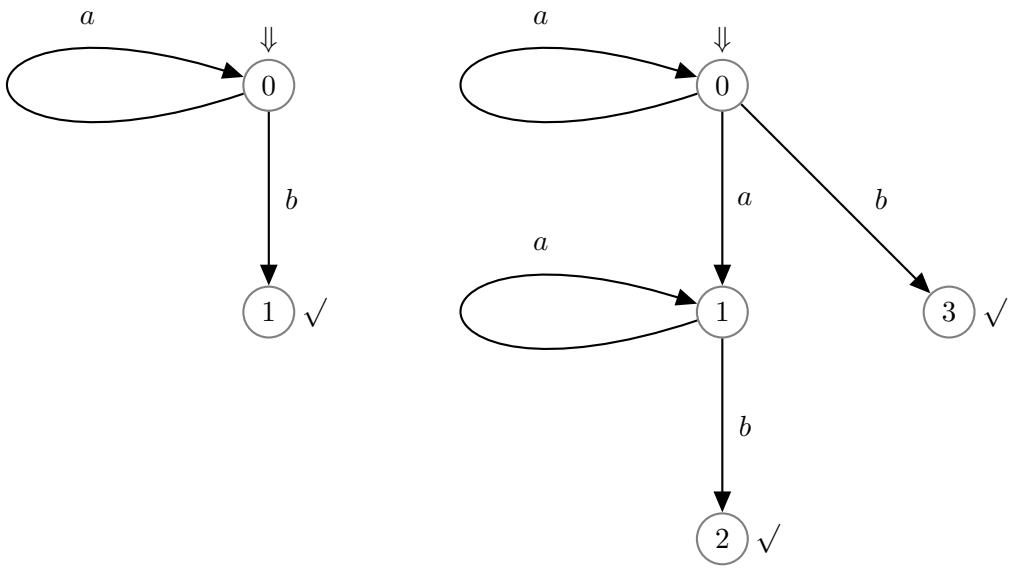
Exercise 6.22 Find a PDL formula that can distinguish between the root states of the following two process graphs:





The formula should be true in one graph, false in the other.

Exercise 6.23 Now consider the following two pictures of process graphs:



Is it still possible to find a PDL formula that is true in the root of one of the graphs and false in the root of the other? If your answer is 'yes', then give such a formula. If your answer is 'no', then try to explain as clearly as you can why you think this is impossible.

6.7 Semantics of PDL

The formulas of PDL are interpreted in states of a labeled transition system (or: LTS, or: process graph), and the actions a of PDL as binary relations on the domain S of the LTS. We can think of an LTS as given by its set of states S, its valuation V, and its set of labelled transitions R. We will give the interpretation of basic actions a as $\xrightarrow{a}$.

If an LTS M is given, we use S_M to refer to its set of states, we use R_M to indicate its set of labelled transitions, and we use V_M for its valuation.

Definition 6.24 (Semantics of PDL) Given is a labelled transition system $M = \langle S, V, R\rangle$ for P and A.

$$
\begin{array}{lcl}
M, s \models \top & & \text{always} \\
M, s \models p & \iff & p \in V(s) \\
M, s \models \neg\varphi & \iff & M, s \not\models \varphi \\
M, s \models \varphi \vee \psi & \iff & M, s \models \varphi \text{ or } M, s \models \psi \\
M, s \models \varphi \wedge \psi & \iff & M, s \models \varphi \text{ and } M, s \models \psi \\
M, s \models \langle\alpha\rangle\varphi & \iff & \text{for some } t, (s,t) \in [\![\alpha]\!]^M \text{ and } M, t \models \varphi \\
M, s \models [\alpha]\varphi & \iff & \text{for all } t \text{ with } (s,t) \in [\![\alpha]\!]^M \text{ it holds that } M, t \models \varphi.
\end{array}
$$

where the binary relation $[\![\alpha]\!]^M$ interpreting the action α in the model M is defined as

$$\begin{aligned}
[\![a]\!]^M &= \xrightarrow{a}_M \\
[\![?\varphi]\!]^M &= \{(s,s) \in S_M \times S_M \mid M, s \models \varphi\} \\
[\![\alpha_1;\alpha_2]\!]^M &= [\![\alpha_1]\!]^M \circ [\![\alpha_2]\!]^M \\
[\![\alpha_1 \cup \alpha_2]\!]^M &= [\![\alpha_1]\!]^M \cup [\![\alpha_2]\!]^M \\
[\![\alpha^*]\!]^M &= ([\![\alpha]\!]^M)^*
\end{aligned}$$

Note that the clause for $[\![\alpha^*]\!]^M$ uses the definition of reflexive transitive closure that was given on page 6-16.

These clauses specify how formulas of PDL can be used to make assertions about PDL models.

Example 6.25 The formula $\langle a \rangle \top$, when interpreted at some state in a PDL model, expresses that that state has a successor in the $\xrightarrow{a}$ relation in that model.

A PDL formula φ is *true* in a model if it holds at every state in that model, i.e., if $[\![\varphi]\!]^M = S_M$.

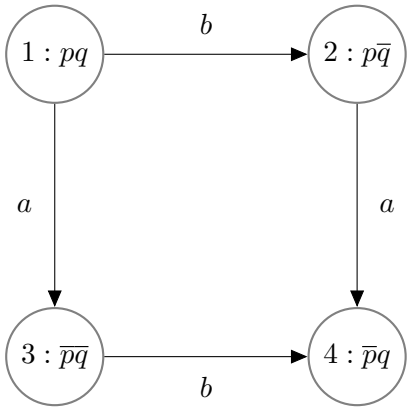
Example 6.26 Truth of the formula $\langle a \rangle \top$ in a model expresses that $\xrightarrow{a}$ is serial in that model. (A binary relation R is serial on a domain S if it holds for all $s \in S$ that there is some $t \in S$ with sRt.)

A PDL formula φ is *valid* if it holds for all PDL models M that φ is true in that model, i.e., that $[\![\varphi]\!]^M = S_M$.

Exercise 6.27 Show that $\langle a;b \rangle \top \leftrightarrow \langle a \rangle \langle b \rangle \top$ is an example of a valid formula.

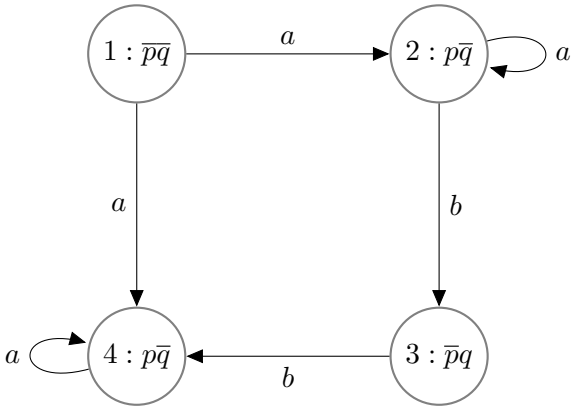
As was note before, ? is an operation for mapping formulas to action statements. Action statements of the form $?\varphi$ are called *tests*; they are interpreted as the identity relation, restricted to the states satisfying the formula.

Exercise 6.28 Let the following PDL model be given:



Give the interpretations of $?p$, of $?(p \vee q)$, of $a; b$ and of $b; a$.

Exercise 6.29 Let the following PDL model be given:



(1) List the states where the following formulas are true:

a. $\neg p$

b. $\langle b \rangle q$

c. $[a](p \rightarrow \langle b \rangle q)$

(2) Give a formula that is true only at state 4.

(3) Give all the elements of the relations defined by the following action expressions:

a. $b; b$

b. $a \cup b$

c. a^*

(4) Give a PDL action expression that defines the relation $\{(1, 3)\}$ in the graph. (Hint: use one or more test actions.)

Converse Let $\breve{}$ (converse) be an operator on PDL programs with the following interpretation:

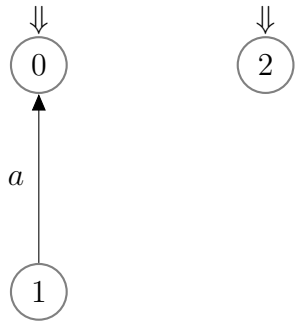
$$[\![\alpha\breve{}]\!]^M \quad = \quad \{(s, t) \mid (t, s) \in [\![\alpha]\!]^M\}.$$

Exercise 6.30 Show that the following equalities hold:

$$\begin{aligned} (\alpha; \beta)\breve{} &= \beta\breve{}; \alpha\breve{} \\ (\alpha \cup \beta)\breve{} &= \alpha\breve{} \cup \beta\breve{} \\ (\alpha^*)\breve{} &= (\alpha\breve{})^* \end{aligned}$$

Exercise 6.31 Show how the equalities from the previous exercise, plus atomic converse $a\breve{}$, can be used to define $\alpha\breve{}$, for arbitrary α, by way of abbreviation.

It follows from Exercises 6.30 and 6.31 that it is enough to add converse to the PDL language for atomic actions only. To see that adding converse in this way increases expressive power, observe that in root state 0 in the following picture $\langle a^{\smile}\rangle\top$ is true, while in root state 2 in the picture $\langle a^{\smile}\rangle\top$ is false. On the assumption that 0 and 2 have the same valuation, no PDL formula without converse can distinguish the two states.



6.8 Axiomatisation

The logic of PDL is axiomatised as follows. Axioms are all propositional tautologies, plus an axiom stating that α behaves as a standard modal operator, plus axioms describing the effects of the program operators (we give box ($[\alpha]$)versions here, but every axiom has an equivalent diamond ($\langle\alpha\rangle$) version), plus a propositional inference rule and a modal inference rule.

The propositional inference rule is the familiar rule of Modus Ponens.

(modus ponens) From $\vdash \varphi_1$ and $\vdash \varphi_1 \to \varphi_2$, infer $\vdash \varphi_2$.

The modal inference rule is the rule of modal generalization (or: necessitation):

(modal generalisation) From $\vdash \varphi$, infer $\vdash [\alpha]\varphi$.

Modal generalization expresses that theorems of the system have to hold in every state.

Example 6.32 Take the formula $(\varphi \wedge \psi) \to \varphi$. Because this is a propositional tautology, it is a theorem of the system, so we have $\vdash (\varphi \wedge \psi) \to \varphi$. And because it is a theorem, it has to hold everywhere, so we have, for any α:

$$\vdash [\alpha]((\varphi \wedge \psi) \to \varphi).$$

Now let us turn to the axioms. The first axiom is the K axiom (familiar from Chapter 5) that expresses that program modalities distribute over implications:

$$\text{(K)} \ \vdash \quad [\alpha](\varphi \to \psi) \to ([\alpha]\varphi \to [\alpha]\psi)$$

Example 6.33 As an example of how to play with this, we derive the equivalent $\langle\alpha\rangle$ version. By the K axiom, the following is a theorem (just replace ψ by $\neg\psi$ everywhere in the axiom):

$$\vdash [\alpha](\varphi \to \neg\psi) \to ([\alpha]\varphi \to [\alpha]\neg\psi).$$

From this, by the propositional reasoning principle of contraposition:

$$\vdash \neg([\alpha]\varphi \to [\alpha]\neg\psi) \to \neg[\alpha](\varphi \to \neg\psi).$$

From this, by propositional reasoning:

$$\vdash [\alpha]\varphi \wedge \neg[\alpha]\neg\psi) \to \neg[\alpha](\varphi \to \neg\psi).$$

Now replace all boxes by diamonds, using the abbreviation $\neg\langle\alpha\rangle\neg\varphi$ for $[\alpha]\varphi$:

$$\vdash \neg\langle\alpha\rangle\neg\varphi \wedge \neg\neg\langle\alpha\rangle\neg\neg\psi) \to \neg\neg\langle\alpha\rangle\neg(\varphi \to \neg\psi).$$

This can be simplified by propositional logic, and we get:

$$\vdash (\neg\langle\alpha\rangle\neg\varphi \wedge \langle\alpha\rangle\psi) \to \langle\alpha\rangle(\varphi \wedge \psi).$$

Example 6.34 This example is similar to Example 5.45 from Chapter 5.

Above, we have seen that $[\alpha]((\varphi \wedge \psi) \to \varphi)$ is a theorem. With the K axiom, we can derive from this:

$$\vdash [\alpha](\varphi \wedge \psi) \to [\alpha]\varphi.$$

In a similar way, we can derive:

$$\vdash [\alpha](\varphi \wedge \psi) \to [\alpha]\psi.$$

From these by propositional reasoning:

$$\vdash [\alpha](\varphi \wedge \psi) \to ([\alpha]\varphi \wedge [\alpha]\psi). \tag{*}$$

The implication in the other direction is also derivable, as follows:

$$\vdash \varphi \to (\psi \to (\varphi \wedge \psi)),$$

because $\varphi \to (\psi \to (\varphi \wedge \psi))$ is a propositional tautology. By modal generalization (necessitation) from this:

$$\vdash [\alpha](\varphi \to (\psi \to (\varphi \wedge \psi))).$$

By two applications of the K axiom and propositional reasoning from this:

$$\vdash [\alpha]\varphi \to ([\alpha]\psi \to [\alpha](\varphi \wedge \psi)).$$

Since $\varphi \to (\psi \to \chi)$ is propositionally equivalent to $(\varphi \wedge \psi) \to \chi$, we get from this by propositional reasoning:

$$\vdash ([\alpha]\varphi \wedge [\alpha]\psi) \to [\alpha](\varphi \wedge \psi). \tag{**}$$

Putting the two principles $(*)$ and $(**)$ together we get:

$$\vdash [\alpha](\varphi \wedge \psi) \leftrightarrow ([\alpha]\varphi \wedge [\alpha]\psi). \tag{***}$$

Let us turn to the next axiom, the axiom for test. This axiom says that $[?\varphi_1]\varphi_2$ expresses an implication:

$$\text{(test)} \ \vdash \quad [?\varphi_1]\varphi_2 \leftrightarrow (\varphi_1 \rightarrow \varphi_2)$$

The axioms for sequence and for choice:

$$\begin{aligned} \text{(sequence)} \ &\vdash \quad [\alpha_1;\alpha_2]\varphi \leftrightarrow [\alpha_1][\alpha_2]\varphi \\ \text{(choice)} \ &\vdash \quad [\alpha_1 \cup \alpha_2]\varphi \leftrightarrow [\alpha_1]\varphi \wedge [\alpha_2]\varphi \end{aligned}$$

Example 6.35 As an example application, we derive

$$\vdash [\alpha;(\beta \cup \gamma)]\varphi \leftrightarrow [\alpha][\beta]\varphi \wedge [\alpha][\gamma]\varphi.$$

Here is the derivation:

$$\begin{array}{lll} [\alpha;(\beta \cup \gamma)]\varphi & \leftrightarrow \text{(sequence)} & [\alpha][\beta \cup \gamma]\varphi \\ & \leftrightarrow \text{(choice)} & [\alpha]([\beta]\varphi \wedge [\gamma]\varphi) \\ & \leftrightarrow (***) & [\alpha][\beta]\varphi \wedge [\alpha][\gamma]\varphi. \end{array}$$

These axioms together reduce PDL formulas without $*$ to formulas of multi-modal logic (propositional logic extended with simple modalities $[a]$ and $\langle a\rangle$).

Example 6.36 We show how this reduction works for the formula $[(a;b) \cup (?\varphi;c)]\psi$:

$$\begin{array}{lll} [(a;b) \cup (?\varphi;c)]\psi & \leftrightarrow \text{(choice)} & [a;b]\psi \wedge [?\varphi;c]\psi \\ & \leftrightarrow \text{(sequence)} & [a][b]\psi \wedge [?\varphi][c]\psi \\ & \leftrightarrow \text{(test)} & [a][b]\psi \wedge (\varphi \rightarrow [c]\psi). \end{array}$$

For the * operation there are two axioms:

$$\begin{aligned} \text{(mix)} \ &\vdash \quad [\alpha^*]\varphi \leftrightarrow \varphi \wedge [\alpha][\alpha^*]\varphi \\ \text{(induction)} \ &\vdash \quad (\varphi \wedge [\alpha^*](\varphi \rightarrow [\alpha]\varphi)) \rightarrow [\alpha^*]\varphi \end{aligned}$$

The mix axiom expresses the fact that α^* is a reflexive and transitive relation containing α, and the axiom of induction captures the fact that α^* is the *least* reflexive and transitive relation containing α.

As was mentioned before, all axioms have dual forms in terms of $\langle\alpha\rangle$, derivable by propositional reasoning. For example, the dual form of the test axiom reads

$$\vdash \langle?\varphi_1\rangle\varphi_2 \leftrightarrow (\varphi_1 \wedge \varphi_2).$$

The dual form of the induction axiom reads

$$\vdash \langle\alpha^*\rangle\varphi \rightarrow \varphi \vee \langle\alpha^*\rangle(\neg\varphi \vee \langle\alpha\rangle\varphi).$$

Exercise 6.37 Give the dual form of the mix axiom.

We will now show that in the presence of the other axioms, the induction axiom is equivalent to the so-called loop invariance rule:

$$\frac{\varphi \to [\alpha]\varphi}{\varphi \to [\alpha^*]\varphi}$$

Here is the theorem:

Theorem 6.38 In PDL without the induction axiom, the induction axiom and the loop invariance rule are interderivable.

Proof. For deriving the loop invariance rule from the induction axiom, assume the induction axiom. Suppose

$$\vdash \varphi \to [\alpha]\varphi.$$

Then by modal generalisation:

$$\vdash [\alpha^*](\varphi \to [\alpha]\varphi).$$

By propositional reasoning we get from this:

$$\vdash \varphi \to (\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)).$$

From this by the induction axiom and propositional reasoning:

$$\vdash \varphi \to [\alpha^*]\varphi.$$

Now assume the loop invariance rule. We have to establish the induction axiom. By the mix axiom and propositional reasoning:

$$\vdash (\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)) \to [\alpha]\varphi.$$

Again from the mix axiom and propositional reasoning:

$$\vdash (\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)) \to [\alpha][\alpha^*](\varphi \to [\alpha]\varphi).$$

From the two above, with propositional reasoning using (***):

$$\vdash (\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)) \to [\alpha](\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)).$$

Applying the loop invariance rule to this yields:

$$\vdash (\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)) \to [\alpha^*](\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)).$$

From this we get the induction axiom by propositional reasoning:

$$\vdash (\varphi \wedge [\alpha^*](\varphi \to [\alpha]\varphi)) \to [\alpha^*]\varphi.$$

This ends the proof. □

Axioms for Converse Suitable axioms to enforce that $a^{\smile}$ behaves as the converse of a are the following:

$$\begin{aligned} \vdash \varphi &\rightarrow [a]\langle a^{\smile}\rangle\varphi \\ \vdash \varphi &\rightarrow [a^{\smile}]\langle a\rangle\varphi \end{aligned}$$

Exercise 6.39 Show that the axioms for converse are sound, by showing that they hold in any state in any LTS.

6.9 Expressive power: defining programming constructs

The language of PDL is powerful enough to express conditional statements, fixed loop statements, and condition-controlled loop statements as PDL programs. More precisely, the conditional statement

$$\text{if } \varphi \text{ then } \alpha_1 \text{ else } \alpha_2$$

can be viewed as an abbreviation of the following PDL program:

$$(?\varphi;\alpha_1) \cup (?\neg\varphi;\alpha_2).$$

The fixed loop statement

$$\text{do } n \text{ times } \alpha$$

can be viewed as an abbreviation of

$$\underbrace{\alpha;\cdots;\alpha}_{n \text{ times}}$$

The condition-controlled loop statement

$$\text{while } \varphi \text{ do } \alpha$$

can be viewed as an abbreviation of

$$(?\varphi;\alpha)^*;?\neg\varphi.$$

This loop construction expressed in terms of reflexive transitive closure works for *finite* repetitions only, for note that the interpretation of " while $\top$ do α" in any model is the empty relation. Successful execution of every program we are considering here involves *termination* of the program.

The condition controlled loop statement

$$\text{repeat } \alpha \text{ until } \varphi$$

can be viewed as an abbreviation of

$$\alpha;(?\neg\varphi;\alpha)^*;?\varphi.$$

Note how these definitions make the difference clear between the *while* and *repeat* statements. A *repeat* statement always executes an action at least once, and next keeps on performing the action until the stop condition holds. A *while* statement checks a continue condition and keeps on performing an action until that condition does not hold anymore. If *while* φ *do* α gets executed, it may be that the α action does not even get executed once. This will happen if φ is false in the start state.

In imperative programming, we also have the *skip* program (the program that does nothing) and the *abort* program (the program that always fails): *skip* can be defined as $?\top$ (this is a test that always succeeds) and *abort* as $\bot$ (this is a test that always fails).

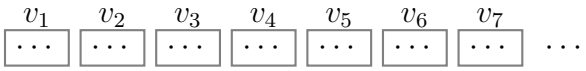
Taking stock, we see that with the PDL action operations we can define the whole repertoire of imperative programming constructs: inside of PDL there is a full fledged imperative programming language.

Moreover, given a PDL program α, the program modalities $\langle\alpha\rangle\varphi$ and $[\alpha]\varphi$ can be used to describe so-called *postconditions of execution* for program α. The first of these expresses that α has a successful exection that ends in an φ state; the second one expresses that every successful execution of α ends in a φ state. We will say more about the use of this in Section 6.10 below.

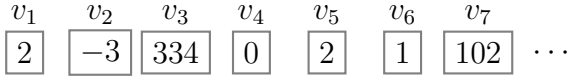
6.10 Outlook — Programs and Computation

If one wishes to interpret PDL as a logic of computation, then a natural choice for interpreting the basic actions statements is as *register assignment statements*. If we do this, then we effectively turn the action statement part of PDL into a very expressive programming language.

Let v range over a set of registers or memory locations V. A V-memory is a set of storage locations for integer numbers, each labelled by a member of V. Let $V = \{v_1, \ldots, v_n\}$. Then a V-memory can be pictured like this:



A V-state s is a function $V \to \mathbb{Z}$. We can think of a V-state as a V-memory together with its contents. In a picture:



If s is a V-state, $s(v)$ gives the contents of register v in that state. So if s is the state above, then $s(v_2) = -3$.

Let i range over integer names, such as 0, -234 or 53635 and let v range over V. Then the following defines arithmetical expressions:

$$a \quad ::= \quad i \mid v \mid a_1 + a_2 \mid a_1 * a_2 \mid a_1 - a_2.$$

It is clear that we can find out the value $[\![a]\!]_s$ of each arithmetical expression in a given V-state s.

Exercise 6.40 Provide the formal details, by giving a recursive definition of $[\![a]\!]_s$.

Next, assume that basic propositions have the form $a_1 \leq a_2$, and that basic action statements have the form $v := a$. This gives us a programming language for computing with integers as action statement language and a formula language that allows us to express properties of programs.

Determinism To say that program α is deterministic is to say that if α executes successfully, then the end state is uniquely determined by the initial state. In terms of PDL formulas, the following has to hold for every φ:

$$\langle\alpha\rangle\varphi \rightarrow [\alpha]\varphi.$$

Clearly, the basic programming actions $v := a$ are deterministic.

Termination To say that program α terminates (or: halts) in a given initial state is to say that there is a successful execution of α from the current state. To say that α always terminates is to say that α has a successful execution from *any* initial state. Here is a PDL version:

$$\langle\alpha\rangle\top.$$

Clearly, the basic programming actions $v := a$ always terminate.

Non-termination of programs comes in with loop constructs. Here is an example of a program that never terminates:

$$\textit{while } \top \textit{ do } v := v + 1.$$

One step through the loop increments the value of register v by 1. Since the loop condition will remain true, this will go on forever.

In fact, many more properties beside determinism and termination can be expressed, and in a very systematic way. We will give some examples of the style of reasoning involved.

Hoare Correctness Reasoning Consider the following problem concerning the outcome of a pebble drawing action.

> A vase contains 35 white pebbles and 35 black pebbles. Proceed as follows to draw pebbles from the vase, as long as this is possible. Every round, draw two pebbles from the vase. If they have the same colour, then put a black pebble into the vase (you may assume that there are enough additional black pebbles outside of the vase). If they have different colours, then put the white pebble back. In every round one pebble is removed from the vase, so after 69 rounds there is a single pebble left. What is the colour of this pebble?

It may seem that the problem does not provide enough information for a definite answer, but in fact it does. The key to the solution is to discover an appropriate *loop invariant*: a property that is initially true, and that does not change during the procedure.

Exercise 6.41 Consider the property: 'the number of white pebbles is odd'. Obviously, this is initially true. Show that the property is a loop invariant of the pebble drawing procedure. What follows about the colour of the last pebble?

It is possible to formalize this kind of reasoning about programs. This formalization is called Hoare logic. One of the seminal papers in computer science is Hoare's [Hoa69]. where the following notation is introduced for specifying what a computer program written in an imperative language (like C or Java) does:

$$\{P\} \quad C \quad \{Q\}.$$

Here C is a program from a formally defined programming language for imperative programming, and P and Q are conditions on the programming variables used in C.

Statement $\{P\}\ C\ \{Q\}$ is true if whenever C is executed in a state satisfying P and if the execution of C terminates, then the state in which execution of C terminates satisfies Q. The 'Hoare-triple' $\{P\}\ C\ \{Q\}$ is called a *partial correctness specification*; P is called its *precondition* and Q its *postcondition*. Hoare logic, as the logic of reasoning with such correctness specifications is called, is the precursor of all the dynamic logics known today.

Hoare correctness assertions are expressible in PDL, as follows. If φ, ψ are PDL formulas and α is a PDL program, then

$$\{\varphi\}\ \alpha\ \{\psi\}$$

translates into

$$\varphi \to [\alpha]\psi.$$

Clearly, $\{\varphi\}\ \alpha\ \{\psi\}$ holds in a state in a model iff $\varphi \to [\alpha]\psi$ is true in that state in that model.

The Hoare inference rules can now be derived in PDL. As an example we derive the rule for guarded iteration:

$$\frac{\{\varphi \wedge \psi\}\ \alpha\ \{\psi\}}{\{\psi\}\ \textit{while}\ \varphi\ \textit{do}\ \alpha\ \{\neg\varphi \wedge \psi\}}$$

First an explanation of the rule. The correctness of *while* statements is established by finding a *loop invariant*. Consider the following C function:

```
int square (int n)
{
  int x = 0;
  int k = 0;
  while (k < n) {
    x = x + 2*k + 1;
    k = k + 1;
  }
  return x;
}
```

How can we see that this program correctly computes squares? By establishing a loop invariant:

$$\{x = k^2\}\ \texttt{x = x + 2*k + 1; k = k + 1;}\ \{x = k^2\}.$$

What this says is: if the state before execution of the program is such that $x = k^2$ holds, then in the new state, after execution of the program, with the new values of the registers x and k, the relation $x = k^2$ still holds. From this we get, with the Hoare rule for *while*:

$$\{x = k^2\}$$
$$\texttt{while (k < n) \{ x = x + 2*k + 1; k = k + 1; \}}$$
$$\{x = k^2 \wedge k = n\}$$

Combining this with the initialisation:

$$\{\top\}$$
$$\texttt{int x = 0 ; int k = 0;}$$
$$\{x = k^2\}$$
$$\texttt{while (k < n) \{ x = x + 2*k + 1; k = k + 1; \}}$$
$$\{x = k^2 \wedge k = n\}$$

This establishes that the *while* loop correctly computes the square of n in x.

So how do we derive the Hoare rule for *while* in PDL? Let the premise $\{\varphi \wedge \psi\}\ \alpha\ \{\psi\}$ be given, i.e., assume (6.1).

$$\vdash (\varphi \wedge \psi) \rightarrow [\alpha]\psi. \tag{6.1}$$

We wish to derive the conclusion

$$\vdash \{\psi\}\ \textit{while}\ \varphi\ \textit{do}\ \alpha\ \{\neg\varphi \wedge \psi\},$$

i.e., we wish to derive (6.2).

$$\vdash \psi \rightarrow [(?\varphi; \alpha)^*; ?\neg\varphi](\neg\varphi \wedge \psi). \tag{6.2}$$

From (6.1) by means of propositional reasoning:

$$\vdash \psi \to (\varphi \to [\alpha]\psi).$$

From this, by means of the test and sequence axioms:

$$\vdash \psi \to [?\varphi;\alpha]\psi.$$

Applying the loop invariance rule gives:

$$\vdash \psi \to [(?\varphi;\alpha)^*]\psi.$$

Since ψ is propositionally equivalent with $\neg\varphi \to (\neg\varphi \wedge \psi)$, we get from this by propositional reasoning:

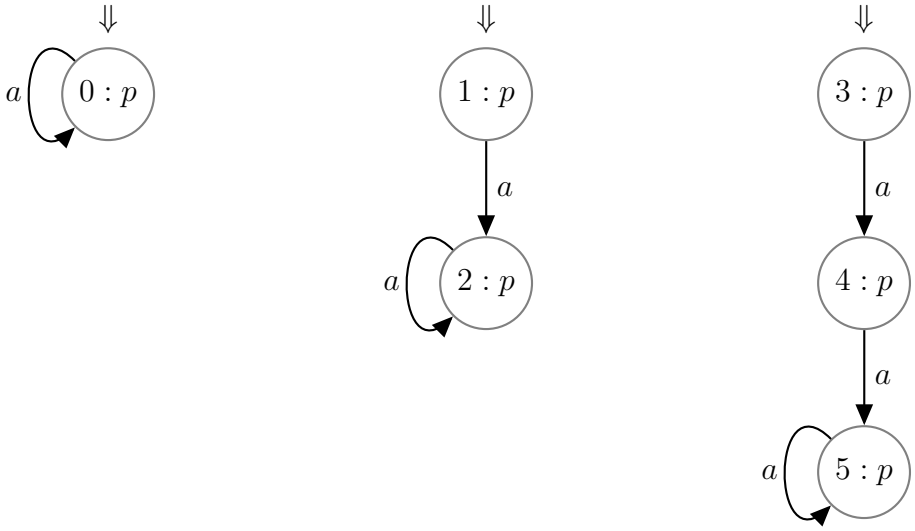
$$\vdash \psi \to [(?\varphi;\alpha)^*](\neg\varphi \to (\neg\varphi \wedge \psi)).$$

The test axiom and the sequencing axiom yield the desired result (6.2).

6.11 Outlook — Equivalence of Programs and Bisimulation

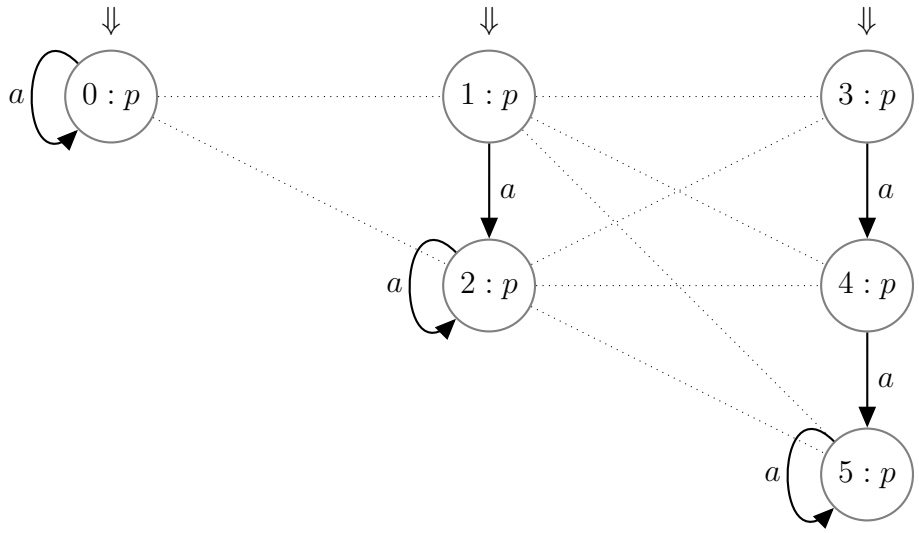
PDL is interpreted in labelled transition systems, and labelled transition systems represent processes. But the correspondence between labelled transition systems and processes is not one-to-one.

Example 6.42 The process that produces an infinite number of a transitions and nothing else can be represented as a labelled transition system in lots of different ways. The following representations are all equivalent, and all represent that process. We further assume that some atomic proposition p is true in all states in all structures.



Each of these three process graphs pictures what is intuitively the following process: that of repeatedly doing a steps, while remaining in a state satisfying p, with no possibility of escape. Think of the actions as ticks of clock, and the state as the state of being imprisoned. The clock ticks on, and you remain in jail forever.

It does not make a difference *for what we can observe directly* (in the present case: that we are in a p state) and *for what we can do* (in the present case: an a action, and nothing else) whether we are in state 0, 1, 2, 3, 4 or 5. From a local observation and action perspective, all of these states are *equivalent*. Below we will make this notion of equivalence precise. For now, we indicate it with connecting lines, as follows:



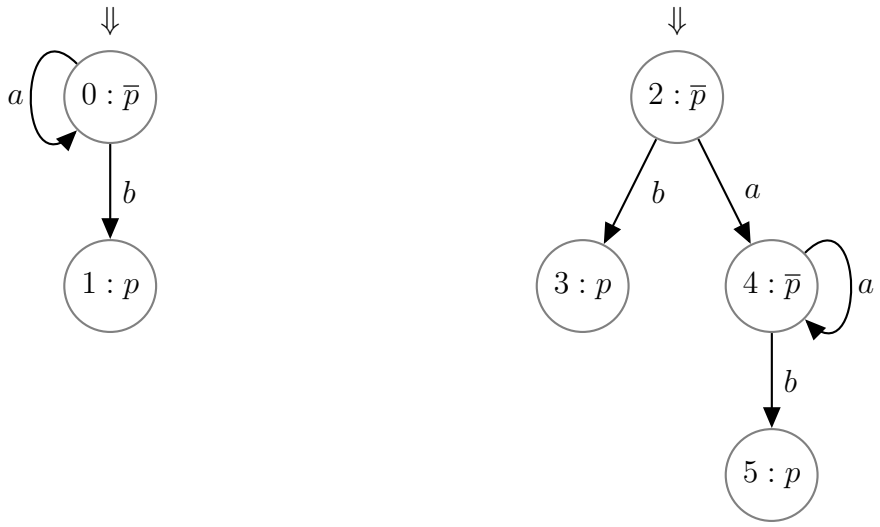
To connect the example to PDL: in all states in each process graph the formulas $\langle a^* \rangle p$, $\langle a; a^* \rangle p$, $\langle a; a; a^* \rangle p$, and so on, are all true. Moreover, it will not be possible to find a PDL formula that sees a difference between the root states of the three process graphs.

We will give a formal definition of this important relation of 'being equivalent from a local action perspective'. We call this relation *bisimulation*, and we say that states that are in the relation are *bisimilar*. Common notation for this is the symbol $\underline{\leftrightarrow}$. Thus, $s \mathrel{\underline{\leftrightarrow}} t$ expresses that there is some relation C which is a bisimulation, such that sCt.

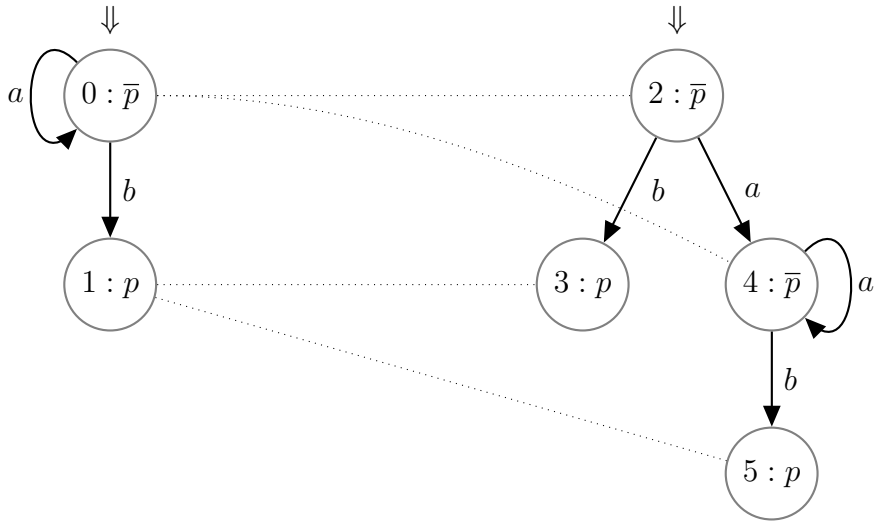
For the picture above we have: $0 \mathrel{\underline{\leftrightarrow}} 1$, $0 \mathrel{\underline{\leftrightarrow}} 2$, and also, between the middle and the right graph: $1 \mathrel{\underline{\leftrightarrow}} 3$, $1 \mathrel{\underline{\leftrightarrow}} 4$, $1 \mathrel{\underline{\leftrightarrow}} 5$, $2 \mathrel{\underline{\leftrightarrow}} 3$, $2 \mathrel{\underline{\leftrightarrow}} 4$, $2 \mathrel{\underline{\leftrightarrow}} 5$. The composition of two bisimulations is again a bisimulation, and we get from the above that we also have: $0 \mathrel{\underline{\leftrightarrow}} 3$, $0 \mathrel{\underline{\leftrightarrow}} 4$ and $0 \mathrel{\underline{\leftrightarrow}} 5$.

We can also have bisimilarity within a single graph: $1 \mathrel{\underline{\leftrightarrow}} 2$, and $3 \mathrel{\underline{\leftrightarrow}} 4$, $3 \mathrel{\underline{\leftrightarrow}} 5$, $4 \mathrel{\underline{\leftrightarrow}} 5$. Note that every node is bisimilar with itself.

Example 6.43 For another example, consider the following picture. Atom p is false in states 0, 2, and 4, and true in states 1, 3 and 5.



In the labelled transition structures of the picture, we have that $0 \underline{\leftrightarrow} 2$, and that $0 \underline{\leftrightarrow} 4$; and $1 \underline{\leftrightarrow} 3$ and $1 \underline{\leftrightarrow} 5$. In a picture:



The notion of *bisimulation* is intended to capture such process equivalences.

Definition 6.44 (Bisimulation) A bisimulation C between LTSs M and N is a relation on $S_M \times S_N$ such that if sCt then the following hold:

Invariance $V_M(s) = V_N(t)$ (the two states have the same valuation),

Zig if for some $s' \in S_M$ $s \xrightarrow{a} s' \in R_M$ then there is a $t' \in S_N$ with $t \xrightarrow{a} t' \in R_N$ and $s'Ct'$.

Zag same requirement in the other direction: if for some $t' \in S_N$ $t \xrightarrow{a} t' \in R_N$ then there is an $s' \in S_M$ with $s \xrightarrow{a} s' \in R_M$ and $s'Ct'$.

The notation $M, s \leftrightarrows N, t$ indicates that there is a bisimulation C that connects s and t. In such a case one says that s and t are bisimilar.

Let M, N be a pair of models and let $C \subseteq S_M \times S_N$. Here is an easy check to see whether C is a bisimulation. For convenience we assume that each model has just a single binary relation (indicated as R_M and R_N). Checking the *invariance* condition is obvious. To check the *zig* condition, check whether

$$C^\smile \circ R_M \subseteq R_N \circ C^\smile.$$

To check the *zag* condition, check whether

$$C \circ R_N \subseteq R_M \circ C.$$

Example 6.45 (Continued from Example 6.43) To see how this works, consider the two models of Example 6.43. Let C be given by

$$\{(0,2),(0,4),(1,3),(1,5)\}.$$

Then the invariance condition holds, for any two states that are C-connected agree in the valuation for p.

Furthermore, $C^\smile \circ R_{M,a} = \{(0,0)\}$ and $R_{N,a} \circ C^\smile = \{(0,0)\}$, so the zig condition holds for the a labels. $C^\smile \circ R_{M,b} = \{(0,1)\}$, and $R_{N,b} \circ C^\smile = \{(0,1)\}$, so the zig condition also holds for the b labels.

Finally, $C \circ R_{N,a} = \{(2,4),(4,4)\}$ and $R_{M,a} \circ C = \{(2,4),(4,4)\}$, so the zag condition holds for the a labels. $C \circ R_{N,b} = \{(0,3),(0,5)\}$, and $R_{M,b} \circ C = \{(0,3),(0,5)\}$, so the zag condition also holds for the b labels.

This shows that C is a bisimulation.

Exercise 6.46 Have another look at Exercise 6.23. Explain why it is impossible to find a PDL formula that is true at the root of one of the graphs and false at the root of the other graph.

Bisimulation is intimately connected to modal logic and to PDL. Modal logic is a sublogic of PDL. It is given by restricting the set of programs to atomic programs.

$$\varphi \quad ::= \quad \top \mid p \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \langle a \rangle \varphi$$

Modal formulas can be used to define global properties of LTSs, as follows. Any modal formula φ can be viewed as a function that maps an LTS M to a subset of S_M, namely the set of those states where φ is true. Call this set φ_M. A global property φ is *invariant for bisimulation* if whenever C is a bisimulation between M and N with sCt, then $s \in \varphi_M$ iff $t \in \varphi_N$.

The notion of invariance for bisimulation generalises the invariance condition of bisimulations.

Exercise 6.47 Show that all modal formulas are invariant for bisimulation: If φ is a modal formula that is true of a state s, and s is bisimilar to t, then φ is true of t as well. (Hint: use induction on the structure of φ.)

Bisimulations are also intimately connected to PDL. Any PDL program α can be viewed as a global relation on LTSs, for α can be viewed as a function that maps an LTS M to a subset of $S_M \times S_M$, namely, the interpretation of α in M. Call this interpretation α_M. A global relation α is *safe for bisimulation* if whenever C is a bisimulation between M and N with sCt, then:

Zig: if $s\alpha_M s'$ for some $s' \in S_M$ then there is a $t' \in S_N$ with $t\alpha_N t'$ and $s'Ct'$,

Zag: vice versa: if $t\alpha_N t'$ for some $t' \in S_N$ then there is an $s' \in S_M$ with $s\alpha_M s'$ and $s'Ct'$.

The notion of safety for bisimulation generalises the zig and zag conditions of bisimulations.

Exercise 6.48 A modal action is a PDL program (action statement) that does not contain *. Use induction on the structure of α to show that all modal actions α are safe for bisimulation.

Summary of Things You Have Learnt in This Chapter *You have learnt how to look at action in a general way, and how to apply a general formal perspective to the analysis of action. You know what labelled transition systems (or: process graphs) are, and you are able to evaluate PDL formulas in states of LTSs. You understand how key programming concepts such as test, composition, choice, repetition, converse are handled in PDL, and how the familiar constructs 'skip','if-then-else', 'while-do', and 'repeat-until' can be expressed in terms of the PDL operations. You are able to check if a given program can be executed on a simple labelled transition system. Finally, you have an intuitive grasp of the notion of bisimulation, and you are able to check whether two states in a single process graph or in different process graphs are bisimilar.*

Further Reading An influential philosophy of action is sketched in [Dav67]. A classical logic of actions is PDL or propositional dynamic logic [Pra78, Pra80, KP81]. A textbook treatment of dynamic logic is presented in [HKT00].

Precise descriptions of how to perform given tasks are called algorithms. The logic of actions is closely connected to the theory of algorithm design. See [DH04]. Connections between logic and (functional) programming are treated in [DvE04].

Social actions are the topic of [EV09].

Chapter 7

Logic, Games and Interaction

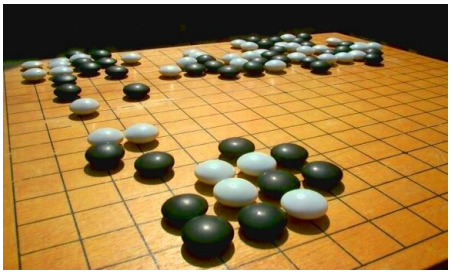
Overview When we bring the logical systems for information and for action from the preceding two chapters together, we get to a ubiquitous phenomenon that has a more "social character": processes of interaction, where different agents respond to each other. Some people think that logic is only about lonely thinkers on their own, but most rational activities involve many agents: think of a case in court, or a process of inquiry in a scientific research group.

This chapter will not present a new logical system, but will demonstrate the fruitfulness of looking at logic from the viewpoint of interaction. We look at argumentation as a game. We give an interactive account of evaluation of assertions in models. We will explain a fundamental result about finite zero-sum two-player games, and we will show you how to apply it. We introduce sabotage games and model comparison games, we explain backward induction and the notion of strategic equilibrium. This brings us into the realm of game theory proper, where we introduce and discuss the basic notions and point out connections with logic.

7.1 Logic meets Games

In many human core activities: conversation, argumentation, but also games that we play in general, social interaction is the heart of the matter, and people often follow rules for their responses over time, called *strategies*. These processes are subtle. In particular, in games, an activity we are all familiar with, strategies are chosen so as to best serve certain goals that the players have, depending on their preferences between different outcomes of the game. Cognitive scientists have argued that what makes us humans so special in the biological world is in fact this social intelligence.





Games fit naturally with the logics developed in this course. Contacts between logic and games go back to Antiquity, since logic arose in a context of argumentation, where valid inferences correspond to successful moves that participants can safely make. We will explain this connection a bit further in what follows, but just think of this. Argumentation is a sort of game, where you have to respond to others following certain rules, where timing of what you bring up matters, and where you can win or lose, depending on the quality of your strategies (and the sensibility of your claims). And here is an attractive idea that has recurred in the history of logic: players who stick to defending logically valid claims have a "winning strategy", a rule that guarantees success in winning debates.

Example 7.1 (Argumentation as a game) Here is an illustration making this a bit more precise. Consider this useful inference that you have encountered many times in Chapter 2:

from premises $\neg\varphi$, $\varphi \vee \psi$ to conclusion ψ.

Here is how we can see this function in an argumentation game.

A Proponent (player P) defends claim ψ against an Opponent (O) who has committed to the premises $\neg\varphi$, $\varphi \vee \psi$. The procedure is one where each player speaks in turn. We record some moves:

1 O starts by challenging P to produce a defense of ψ.

2 P now presses O on one of his commitments. $\varphi \vee \psi$, demanding a choice.

3 O must respond to this, having nothing else to say. There are two options here, which we list separately:

3' O commits to φ.

4' P now points at O's commitment to $\neg\varphi$, and wins because of O's self-contradiction.

3" O commits to ψ.

4" Now P uses this concession to make his own defense to 1. O has nothing further to say, and loses.

You see clearly how logical steps become moves in an argumentation scenario.

But argumentation is only one example of logic meeting games. Nowadays, there are many precise "logic games" for such tasks as evaluation of formulas in models, comparing models for similarity, finding proofs, and many other things of interest. We will discuss a few later, giving you an impression of what might be called *the game of logic*.

But there is more to the interface of logic and games. As we said already, interaction between many agents also involves their preferences, goals, and strategic behaviour where these are as important as the pure information that they have, or obtain. Such richer games have typically been studied, not in logic, but in the field of *game theory* which studies games of any sort: from recreational games to economic behaviour and warfare. Now, one striking recent development is the emergence of connections between logic and game theory, where logics are used to analyze the structure of games, and the reasoning performed by players as they try to do the best they can. The resulting *logics of games* are a natural continuation of the epistemic and dynamic logics that you have seen in the preceding chapters. We will also give you a brief glimpse of this modern link. Actually, this new interface developing today is not just an affair with two partners. It also involves computer science (the area of "agency" which studies complex computer systems plus their users) and philosophy, especially epistemology (the theory of knowledge) and philosophy of action. We will say something about these contacts in the Outlooks at the end.

This chapter will not gang up on you with one more core system that you must learn to work with, the way we have done in previous chapters. Its main aim is to give you an impression of how many earlier logical themes meet naturally in the arena of games, as a sort of combined finale. We start with a series of logical games, that should throw some new light on the logical systems that you have already learnt in this course. Following that, we discuss general games, and what logic has to say about them. All this is meant as a first introduction only. If you want to learn more about these interfaces, some very recent, you should go to a advanced course, or the specialized literature of today.

7.2 Evaluation of Assertions as a Logical Game

Our first example of a logical game is not argumentation, but something even simpler, the semantic notion of truth and falsity for formulas in models, as we have seen it in all our chapters, from propositional and predicate logic to epistemic and dynamic logic. Understanding complex logical expressions may itself be viewed as a game. A historical example is a famous explanation by Leibniz of the basic universal/existential quantifier combination that you have studied in Chapter 4. He did this in terms of two mathematicians discussing a logical formula of the form $\forall x \exists y \varphi(x, y)$. One of the mathematicians says to the other: if you challenge me with an x I will take on your challenge by giving you an y such that φ holds of x and y.

Here is a concrete example: the definition of continuity by Karl Weierstrass (1815–1897):

$$\forall x \forall \epsilon > 0 \exists \delta > 0 \forall y (|x - y| < \delta \rightarrow |f(x) - f(y)| < \epsilon).$$

This formula says something rather complicated about a function f, namely that f is continuous at every point x. The meaning can be unravelled by giving it the form of a dialogue.

Leibniz thought of a mathematician playing the universal quantifier as issuing a "challenge": any object for x. The other player, for the existential quantifier, then must come with an appropriate response, choosing some object for y that makes the assertion $\varphi(x, y)$ true. The following game is a generalization of this idea.

Remark on naming Logical games often have two players with opposing roles. There are many names for them: Players 1 and 2, Abelard and Eloise, Adam and Eve, $\forall$ and $\exists$, Spoiler and Duplicator, Opponent and Proponent, Falsifier and Verifier. In this chapter, we will use a selection from these, or we use the neutral I and II.

For a start, recall the basic semantic notion of predicate logic in Chapter 4, truth of a formula φ in a model $\mathcal{M}$ under an assignment s of objects to variables:

$$\mathcal{M}, s \models \varphi$$

Now, stepwise evaluation of first-order assertions can be understood dynamically as a game of evaluation for two players. Verifier **V** claims that φ is true in the setting $\mathcal{M}, s$, Falsifier **F** that it is false.

Definition 7.2 (Evaluation games) The natural moves of defense and attack in the first-order evaluation game will be indicated henceforth as

$$\text{game}(\varphi, \mathcal{M}, s)$$

The moves of evaluation games follow the inductive construction of formulas. They involve some typical actions that occur in games, such as *choice, switch*, and *continuation*, coming in dual pairs with both players **V** (Verifier) and **F** (Falsifier) allowed the initiative once:

Atoms Pd, Rde, ...
V wins if the atom is true, **F** if it is false

Disjunction $\varphi_1 \vee \varphi_2$:
V chooses which disjunct to play

Conjunction $\varphi_1 \wedge \varphi_2$:
F chooses which conjunct to play

Negation $\neg\varphi$:
Role switch between the players, play continues with respect to φ.

Next, the quantifiers make players look inside $\mathcal{M}$'s domain of objects, and pick objects:

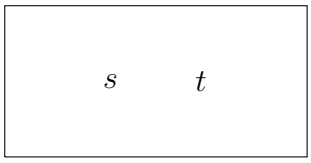
Existential quantifiers $\exists x\varphi(x)$:
V picks an object d, and then play continues with respect to $\varphi(d)$.

Universal quantifiers $\forall x\varphi(x)$:
The same, but now for **F**.

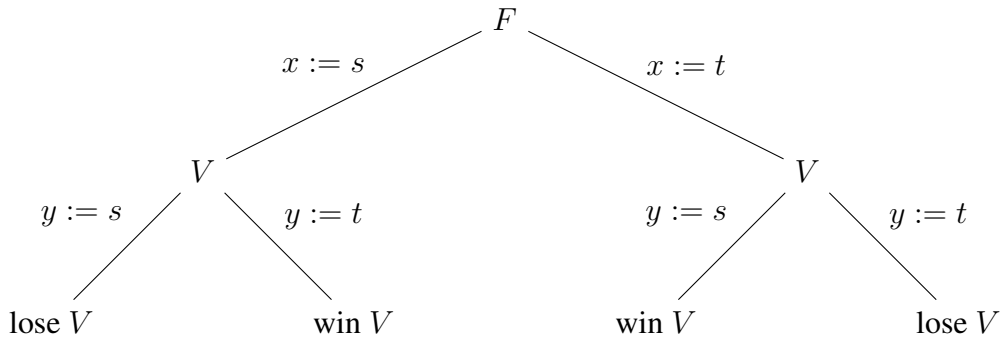
The game ends at atomic formulas: Verifier wins if it is true, Falsifier wins if it is false.

The schedule of the game is determined by the form of the statement φ. To see this in a very simple case, consider the following example.

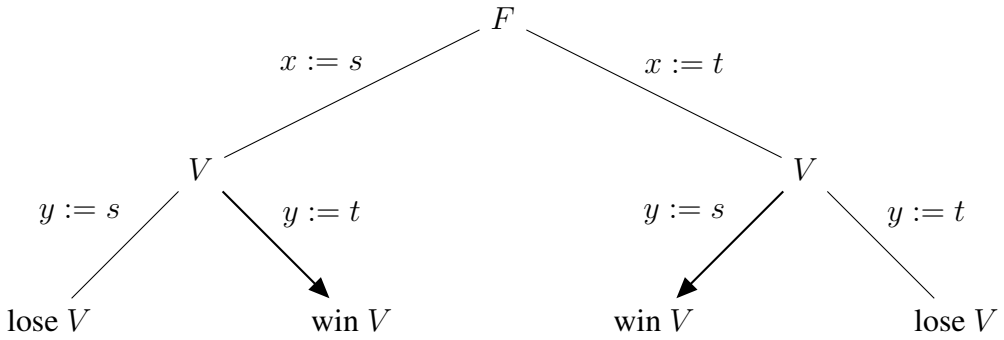
Example 7.3 (Evaluation Game With Two Objects) Let $\mathcal{M}$ be a model with two objects:



Here is the complete game for the first-order formula $\forall x \exists y x \neq y$ as a tree of moves, with scheduling from top to bottom (note that $x := s$ is shorthand for the action of picking object s for x):

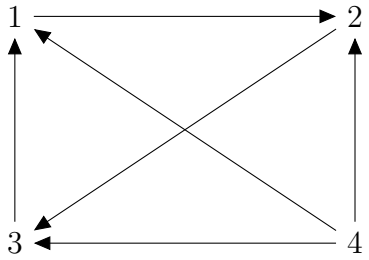


Falsifier starts, Verifier must respond. There are four possible plays, with two wins for each player. But Verifier is the player with a winning strategy, in an obvious sense: she has a rule for playing that will make her win no matter what Falsifier does: "choose the other object". We can indicate this by high-lighting her recommended moves in bold-face:



Evaluation games for complex formulas in richer models can be more challenging.Here is an example going back to the graphs used in Chapters 3, 4 and 5.

Example 7.4 (Find non-communicators!) Consider the following communication network with arrows for directed links, and with all reflexive 'self-loops' present but suppressed for convenience in the drawing:



In this setting, the predicate-logical formula

$$\forall x \forall y (Rxy \vee \exists z (Rxz \wedge Rzy))$$

claims that every two nodes in this network can communicate in at most two steps. Here is a possible run of the corresponding evaluation game:

F picks 2, game continues for $\forall y (R2y \vee \exists z (R2z \wedge Rzy))$

F picks 1, game continues for $(R21 \vee \exists z (R2z \wedge Rz1))$

V chooses $\exists z (R2z \wedge Rz1)$

V picks 4, game continues for $(R24 \wedge R41)$

F chooses $R41$.

test: V wins.

In this run, Falsifier started off with a threat by picking object 2, but then became generous towards Verifier, picking object 1. Verifier accepted the present by choosing the true right conjunct, but then tripped up by picking the wrong witness 4 instead of 3. But once again, Falsifier did not exploit this, by choosing the true right-hand conjunct. Obviously, however, Falsifier has a winning strategy in this game, exploiting the 'counter-example' of object 2, which cannot reach 1 in ≤ 2 steps. He even has more than one such strategy, as $x = 2$, $y = 4$ would also serve as a rule that always makes him win.

Exercise 7.5 Every finite network in which distinct points always have at least one directed link contains a 'Great Communicator': an object which can reach every other node in at most 2 steps. Prove this, and describe the general winning strategy for Verifier.

Truth and Verifier's winning strategies In our first example, participants were not evenly matched. Player **V** can always win: after all, she is defending the truth of the matter. More precisely, in the above terms, she has a *winning strategy*. As we said, such a strategy is a map from **V**'s turns to her moves following which guarantees, against any counterplay by **F**, that the game will end in outcomes that are won for V. By contrast, **F** has no winning strategy, as this would contradict **V**'s having one. (Playing two winning strategies against each other yields a contradiction.) Neither does have **F** the opposite power of a 'losing strategy': he cannot force **V** to win. Thus, players' powers of controlling outcomes may be quite different. Here is the fundamental connection between truth and games for evaluation games:

Lemma 7.6 (Success Lemma) The following are equivalent for all $\mathcal{M}, s$, and first-order φ:

(1) $\mathcal{M}, s \models \varphi$

(2) **V** has a winning strategy in game$(\varphi, \mathcal{M}, s)$.

A proof for this equivalence, while not hard at all, is beyond the horizon of this chapter.

Exercise 7.7 Prove the Success Lemma by induction on the construction of predicate-logical formulas. Hint: you will find it helpful to show two things simultaneously: (a) If a formula φ is true in $(\mathcal{M}, s)$, then Verifier has a winning strategy, (b) If a formula φ is false in $(\mathcal{M}, s)$, then Falsifier has a winning strategy.

Exercise 7.8 The above definition of evaluation games can be rephrased as follows. There are two kinds of *atomic games*: (a) testing atomic formulas for truth or falsity, but also an operation of (b) picking some object as a value for a variable. Complex games are then constructed out

of these by means of the following operations: (i) choice between two games, (ii) role switch between the players of the game, and (iii) "sequential composition": first playing one game, and then another. Show that all evaluation games for predicate logical formulas can be defined in this manner. Conversely, can you give a game of this more abstract sort that does not correspond to a predicate-logical formula?

7.3 Zermelo's Theorem and Winning Strategies

Logic games involve broader game-theoretical features. Here is a striking one. Our evaluation games have a simple, but striking feature:

Either Verifier or Falsifier must have a winning strategy!

The reason is simply the logical law of Excluded Middle. In any semantic model, either the given formula φ is true, or its negation is true. By the Truth Lemma then, either **V** has a winning strategy in the game for φ, or **V** has a winning strategy in the game for $\neg\varphi$: i.e.,after a role switch, a winning strategy for **F** in the game for φ. Two-player games in which some player has a winning strategy are called *determined*. The general game-theoretic background of our observation is due to the German set theorist Ernst Zermelo, though it was rediscovered independently by Max Euwe, the Dutch world-champion in Chess (1935–1937).

Ernst Zermelo

Max Euwe

We state it here for two-person "zero-sum" games whose players I, II can only win or lose, and where there is a fixed finite bound on the length of all runs.

Theorem 7.9 All zero-sum two-player games of fixed finite depth are determined.

Proof. Here is a simple algorithm determining the player having the winning strategy at any given node of a game tree of this finite sort. It works bottom-up through the game tree. First, colour those end nodes black that are wins for player I, and colour the other end nodes white, being the wins for II. Then extend this colouring stepwise as follows:

If all children of node n have been coloured already, do one of the following:

(1) if player I is to move, and at least one child is black: colour n black; if all children are white, colour n white;

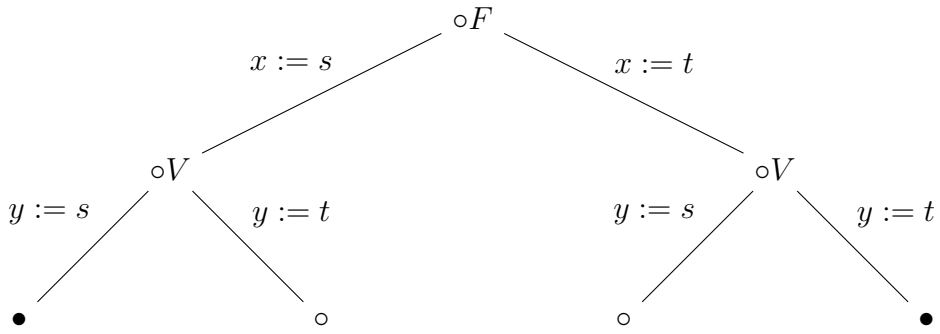
(2) if player II is to move, and at least one child is white: colour n white; if all children are black, colour n black.

This procedure eventually colours all nodes black where player I has a winning strategy, making those where II has a winning strategy white. Here is the reason:

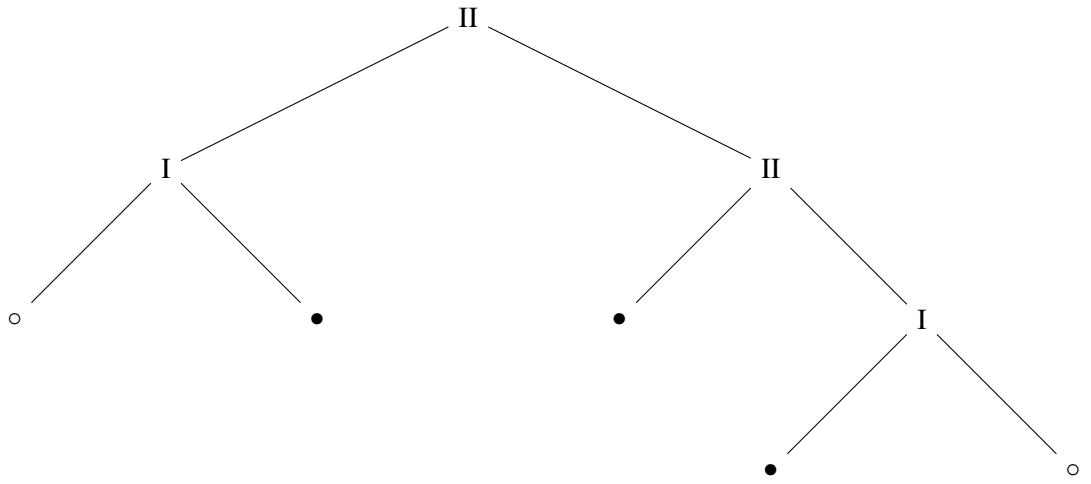
A player has a winning strategy at one of his turns iff he can make a move to at least one daughter node where he has a winning strategy.

□

Here is the correct colouring for the simple game tree of our first example:

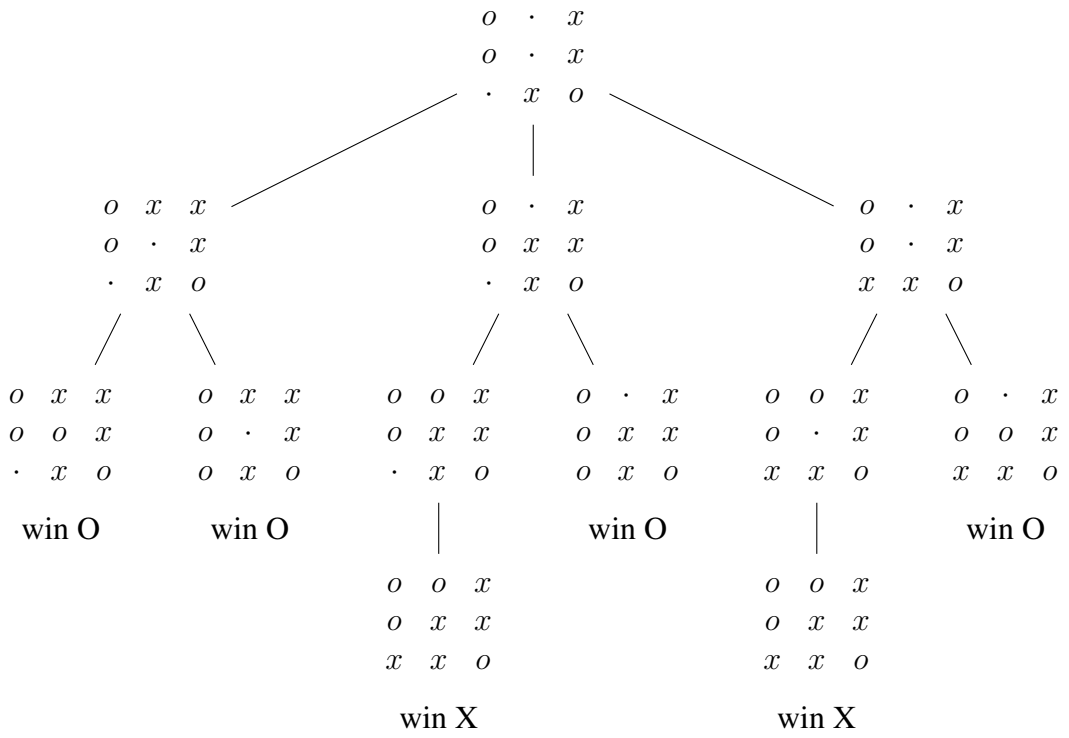


Exercise 7.10 Give the right colouring for the following game, whose black end nodes are wins for player I and white end nodes for player II:



Note how the evaluation games that we defined above satisfy all conditions of Zermelo's Theorem: two players, zero-sum, and finite depth. But its range is much broader. Recursive algorithms like this, in much more sophisticated optimized versions, are widely used in solving real board games, and even general AI search problems.

Example 7.11 Here is part of the game tree for the common game of Noughts and Crosses, indicating all possible moves from the configuration given at the top. It is easy to see that the Nought-player O has a winning strategy there by the colouring algorithm:



Exercise 7.12 Compute all the appropriate colours for the players in this game tree according to the Zermelo algorithm.

Zermelo was mainly concerned with games like chess, which also allow draws. Here the above method implies that one of the two players has a non-losing strategy. The difference

between theory and practice is shown by the following. A century after the original result, it is still unknown which player has a non-losing strategy! But for other highly non-trivial board games, such as Checkers, the Zermelo solution has been found (2007).

Exercise 7.13 Actually, the very proof of Zermelo's Theorem may be cast as a form of Excluded Middle $\varphi \vee \neg\varphi$. Consider a game with 3 moves, and show how the statement of Determinacy can be derived using a suitable first-order formula about players and moves.

Not all two-player games of winning and losing are determined. Counter-examples are games where players need not be able to observe every move made by their opponents, of infinite games, where runs can go on forever.

Exercise 7.14 Consider an infinite game between two players, where histories may go on forever. Using the same style of reasoning as for Zermelo's Theorem, prove the following fact. If player II has no winning strategy at some stage s of the game, then I has a strategy for achieving a set of runs from s during all of which II never has a winning strategy for the remaining game from then on. Explain why this statement is not the same as determinacy for such games.

We hope that we have shown sufficiently how games can be close to the logics that you have learnt, and that thereby, familiar logical laws may acquire striking game-theoretic import. There are many further examples of this interplay, but for that, you will have to go to the literature. For now, we just note that many logical systems have corresponding evaluation games, that are used both as a technical tool, and as a attractive "dynamic" perspective on what logical tasks really are.

Exercise 7.15 Define an evaluation game for the epistemic language of Chapter 5. Hint: Positions of the game will be pointed models (M, s), and the new idea is that modalities move the world s to an accessible successor t. Now specify winning conditions for Verifier and Falsifier in such a way that the Truth Lemma stated above holds for this game with respect to the epistemic language.

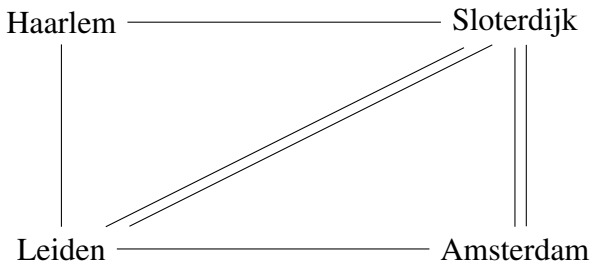
7.4 Sabotage Games: From Simple Actions to Games

This section is a digression. We give one more example of a logic-related game where Zermelo's Theorem plays a role. Our main purpose is to show you how game design is still going on, and you yourself may want to try your hand at it.

The following "sabotage game" was designed, tongue-in-cheek, as a model for railway travel in The Netherlands in periods of strikes and disruptions. Normally, traveling involves solving a search problem "from A to B" along a sequence of links in some fixed network. But what if things get out of hand?

Consider a network consisting of nodes representing cities and links representing ways to travel between them. There are two players: 'Runner' and 'Blocker'. Runner moves from a given starting node A and tries to reach some specified goal node B along existing connections. In each round of the game, Blocker first removes one link from the current network, Runner then moves along one available connection where he is. The game ends if Runner has reached the goal node (he wins then), or if Runner cannot move any more (Blocker then wins).

Example 7.16 In the following railway network, each line is a possible connection for Runner to take:



Runner starts in Haarlem, and wants to reach Amsterdam. Suppose that Blocker first removes the link Haarlem-Sloterdijk. Then Runner can go to Leiden. Now Blocker must remove Leiden-Amsterdam, leaving Runner a link from Leiden to Sloterdijk. Now Blocker is too late: whichever link he cuts between Sloterdijk and Amsterdam, Runner can then use the remaining one to arrive. Does this mean that Runner has a winning strategy in this game? The answer is "No": Blocker has a winning strategy, but it goes as follows.

First cut a link between Sloterdijk and Amsterdam, Then see what Runner does. If he goes to Sloterdijk, cut the second link, and whatever he does next, cut the link Leiden-Amsterdam. If Runner goes to Leiden as his first step, cut the Leiden-Amsterdam link

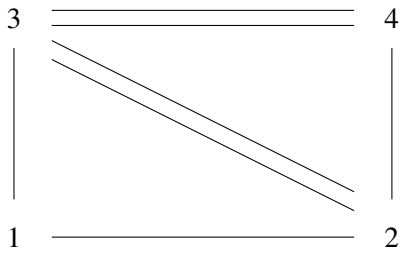
first, then cut the second Sloterdijk-Amsterdam link. Now Amsterdam has become isolated: Runner will never get there.

We have been talking as if the Sabotage Game is determined. And it is, since the conditions of Zermelo's Theorem apply. There are two players, there is just winning and losing as outcomes, and moreover, the game cannot last longer than it takes to cut all links in the given finite graph.

Actually, the Sabotage Game is even closely related to the evaluation games that you have seen before. You can also see it as an evaluation game for a first-order formula on the given graph, which is of the form "For every first move by Blocker, there is a move by Runner in the accessibility relation minus Blockers move such that, for every second move by Blocker . . . etc." Thus, again, logic and games remain close.

Exercise 7.17 Suppose we change the preceding game as follows: Blocker want to *force* Runner to go to Amsterdam, by making it impossible for him to stay anywhere else, assuming that Runner has to move as long as he has open links where he is. By the way, this version has been used to model situations of learning where Teachers are pushing unwilling Students to goal states where they should be. Who has the winning strategy in this new scenario?

Exercise 7.18 Consider the sabotage game with the following initial configuration:



This time, the task for Runner is to start from position 1 and then visit all nodes of the network. Blocker wins if she can somehow prevent this. Who has the winning strategy? How does it work?

You can view the Sabotage Game as a typical multi-agent game version of a standard algorithm for graph search. This is just one instance where the computational perspective of Chapter 6, too, meets with game-theoretic ideas.

7.5 Model Comparison as a Logic Game

Logic games can perform evaluation or argumentation. But they can also be used to perform other basic tasks that you have not learnt about yet in this course. Let us look at one of these, the issue of comparing models. One of the main functions of a language is distinguishing between different situations, represented by models. And one vivid way of

measuring the expressive power of a language is through the following game of spotting differences.

Playing the game Consider any two models $\mathcal{M}$, $\mathcal{N}$. Player D (Duplicator) claims that $\mathcal{M}$, $\mathcal{N}$ are similar, while S (Spoiler) maintains that they are different. Players agree on some finite number k of rounds for the game, 'the severity of the probe'.

Definition 7.19 (Comparison games) A *comparison game* works as follows, packing two moves into one round:

S chooses one of the models, and picks an object d in its domain.

D then chooses an object e in the other model, and the pair (d, e) is added to the current list of matched objects.

At the end of the k rounds, the total object matching obtained is inspected. If this is a 'partial isomorphism', D wins; otherwise, S has won the game.

Here, a *partial isomorphism* is an injective partial function f between models $\mathcal{M}$, $\mathcal{N}$, which is an isomorphism between its own domain and range seen as submodels. This sounds complicated but it is really very easy: a partial isomorphism links finite sets of objects one-to-one in such a way that all their structure is preserved.

Example 7.20 Let $\mathbb{R}$ be the real numbers with the relation 'less than', and let $\mathbb{Q}$ be the rational numbers (the set of all numbers that can be written as p/q, where p and q are integer numbers, with $q \neq 0$). Both of these sets are ordered by $\leq$ ('less than or equal'). Note that 0 can be written as $\frac{0}{1}$, so 0 is a rational number.

For an injective function f between a finite set of reals and a finite set of rationals 'to preserve the structure' in this case means: $x \leq y$ iff $f(x) \leq f(y)$.

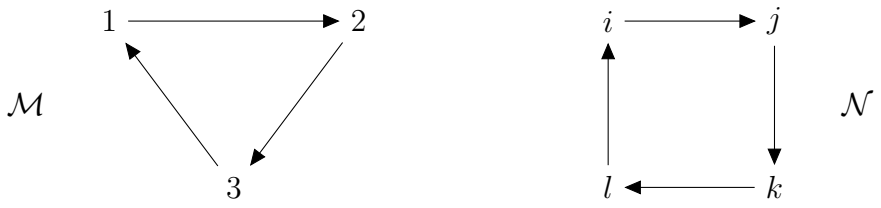
The number $\sqrt{2} \in \mathbb{R}$ is not a fraction. Consider the set of pairs $\{(0, 0), (\sqrt{2}, 1.4142)\}$. This is a partial isomorphism, for it preserves the $\leq$ relation.

Here are some possible runs for models with relations only, as we have often used in Chapter 4, illustrating players' strategies. As before, the game character shows in that players may play badly and lose, but it is their winning strategies that are most important to us. We look at a first-order language with a binary relation symbol R only, mostly disregarding identity atoms with $=$ for the sake of illustration.

Example 7.21 (Playing between graphs: Pin versus Dot) We discuss one run and its implications.

In the first round, **S** chooses a in $\mathcal{M}$, and **D** must choose c in $\mathcal{N}$. If we stopped after one round, **D** would win. There is no detectable difference between single objects in these models. They are all irreflexive, and that's it. But now take a second round. Let **S** choose b in $\mathcal{M}$. Then **D** must again choose c in $\mathcal{N}$. Now **S** wins, as the map $\{(a, c), (b, c)\}$ is not a partial isomorphism. On the lefthand side, there is an R link between a and b, on the righthand side there is none between c and c. Clearly, the structure does not match.

Example 7.22 ('3-Cycle' vs '4-Cycle') Our next example is a match between a '3-Cycle' and a '4-Cycle':



We just display a little table of one possible 'intelligent run':

Round 1 **S** chooses 1 in $\mathcal{M}$, **D** chooses i in $\mathcal{N}$.

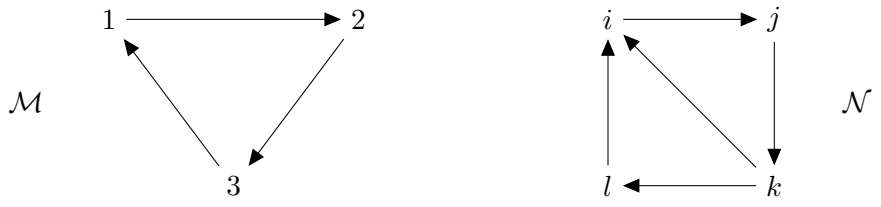
Round 2 **S** chooses 2 in $\mathcal{M}$, **D** chooses j in $\mathcal{N}$.

Round 3 **S** chooses 3 in $\mathcal{M}$, **D** chooses k in $\mathcal{N}$.

S wins, as $\{(1, i), (2, j), (3, k)\}$ is not a partial isomorphism. But he can do even better:

> **S** has a winning strategy in two rounds, first picking i in $\mathcal{N}$, and then taking k in the next round. No such pattern occurs in $\mathcal{M}$, so **D** is bound to lose.

Exercise 7.23 Consider the following variation on the last example.



Which of the two players has a winning strategy in the partial isomorphism game?

Example 7.24 The final example match is 'Integers' $\mathbb{Z}$ versus 'Rationals' $\mathbf{Q}$. These two linear orders have obviously different first-order properties: the latter is dense, the former discrete. Discreteness intuitively means that there are pairs of different numbers with

'nothing in between'. Denseness intuitively means the negation of this: for every pair of different numbers, there is always a number in between. Here is the formal version of density:

$$\forall x \forall y (x < y \to \exists z (x < z \wedge z < y)).$$

And the formal version of discreteness:

$$\exists n \exists m (n < m \wedge \neg \exists k (n < k \wedge k < m)).$$

So this difference between $\mathbb{Q}$ and $\mathbb{Z}$ can be expressed by means of predicate logical formulas. The only question is how soon this will surface in the game.

By choosing his objects well, **D** has a winning strategy here for the game over two rounds. But **S** can always win the game in three rounds. Here is a typical play:

Round 1 **S** chooses 0 in $\mathbb{Z}$, **D** chooses 0 in $\mathbb{Q}$.

Round 2 **S** chooses 1 in $\mathbb{Z}$, **D** chooses $\frac{1}{3}$ in $\mathbb{Q}$.

Round 3 **S** chooses $\frac{1}{5}$ in $\mathbb{Q}$, any response for **D** is losing.

7.6 Different Formulas in Model Comparison Games

Example 7.24 suggests a connection between strategies in model comparison games and formulas of predicate logic. In actual play of model comparison games, you will notice this connection yourself. We discuss it here because it may give you a different perspective on the predicate logic that you have learnt in Chapter 4. In fact, model comparison games throw a lot of new light on predicate logic, as we will explain now.

Winning strategies for **S** are correlated with specific first-order formulas φ that bring out a *difference* between the models. And this correlation is tight. The quantifier syntax of φ triggers the moves for **S**.

Example 7.25 (Continuation of Example 7.21) Exploiting definable differences: 'Pin versus Point'. An obvious difference between the two in first-order logic is

$$\exists x \exists y Rxy$$

Two moves were used by **S** to exploit this, staying inside the model where it holds.

Example 7.26 (Continuation of Example 7.22, '3-Cycle versus 4-Cycle') The first **S**-play exploited the formula

$$\exists x \exists y \exists z (Rxy \wedge Ryz \wedge Rxz)$$

which is true only in $\mathcal{M}$, taking three rounds. The second play, which had only two rounds, used the following first-order formula, which is true only in the model $\mathcal{N}$:

$$\exists x \exists y (\neg Rxy \wedge \neg Ryx \wedge x \neq y).$$

Example 7.27 (Continuation of Example 7.24, 'Integers versus Rationals') **S** might use the definition of density for a binary order that was given above,

$$\forall x \forall y (x < y \rightarrow \exists z (x < z \wedge z < y)),$$

to distinguish $\mathbb{Q}$ from $\mathbb{Z}$. Let us spell this out, to show how the earlier spontaneous play for this example has an almost algorithmic derivation from a first-order difference formula. For convenience, we use density in a form with existential quantifiers only. The idea is for **S** to maintain a difference between the two models, of stepwise decreasing syntactic depth. **S** starts by observing that the negation of density, i.e., the property of discreteness, is true in $\mathbb{Z}$, but false in $\mathbb{Q}$:

$$\exists x \exists y (x < y \wedge \neg \exists z (x < z \wedge z < y)). \qquad (*)$$

He then chooses an integer witness d for x, making

$$\exists y (d < y \wedge \neg \exists z (d < z \wedge z < y))$$

true in $\mathbb{Z}$. **D** can then take any object d' she likes in $\mathbb{Q}$:

$$\exists y (d' < y \wedge \neg \exists z (d' < z \wedge z < y))$$

will always be false for it, by the fact that (*) is false in $\mathbb{Q}$. We have:

$$\mathbb{Z} \models \exists y (d < y \wedge \neg \exists z (d < z \wedge z < y)),\ \mathbb{Q} \not\models \exists y (d' < y \wedge \neg \exists z (d' < z \wedge z < y)).$$

In the second round, **S** continues with a witness e for the new outermost quantifier $\exists y$ in the true existential formula in $\mathbb{Z}$: making $(d < e \wedge \neg \exists z (d < z \wedge z < e))$ true there. Again, whatever object e' **D** now picks in $\mathbb{Q}$, the formula $(d' < e' \wedge \neg \exists z (d' < z \wedge z < e'))$ is false there. In the third round, **S** analyzes the mismatch in truth value. If **D** kept $d' <' e$ true in $\mathbb{Q}$, then, as $\neg \exists z (d < z \wedge z < e))$ holds in $\mathbb{Z}$, $\exists z (d' < z \wedge z < e'))$ holds in $\mathbb{Q}$. **S** then switches to $\mathbb{Q}$, chooses a witness for the existential formula, and wins.

Thus, even the model switches for **S** are encoded in the difference formulas. These are mandatory whenever there is a switch in type from one outermost quantifier to a lower one. Thus, you see how the game is tightly correlated with the structure of the logical language.

Adequacy in terms of quantifier depth Our examples may have suggested the following correlation to you:

> 'winning strategy for **S** over n rounds' versus 'difference formula with n quantifiers'.

But that is not quite the right measure, if you think of Spoiler's reasoning in the above examples. The correct syntactic correlation for the number of rounds needed to win is *syntactic quantifier depth*, being the *maximum length of a quantifier nesting in a formula*. Here is the result that ties it all together.

> Let us write
>
> $$\text{WIN}(\mathbf{D}, \mathcal{M}, \mathcal{N}, k)$$
>
> for: **D** has a *winning strategy* against **S** in the k-round comparison game between the models $\mathcal{M}$ and $\mathcal{N}$.

Comparison games can start from any given 'handicap', i.e., an initial matching of objects in $\mathcal{M}$ and $\mathcal{N}$. In particular, if models have distinguished objects named by individual constants, then these must be matched automatically at the start of the game. In the proofs to follow, for convenience, we will think of all 'initial matches' in the latter way. Now here is the analogue of the Success Lemma for our earlier evaluation games:

Theorem 7.28 (Adequacy Theorem) For all models $\mathcal{M}$, $\mathcal{N}$, all $k \in \mathbb{N}$, the following two assertions are equivalent:

(1) $\text{WIN}(\mathbf{D}, \mathcal{M}, \mathcal{N}, k)$: **D** has a winning strategy in the k-round game.

(2) $\mathcal{M}, \mathcal{N}$ agree on all first-order sentences up to quantifier depth k.

Again, a proof is not hard, but it goes beyond this course. If you go through such a proof (there is of course no harm in trying, it works by induction on the number k), you will find that the situation is even more interesting. There is an explicit correspondence between

(1) winning strategies for **S** in the k-round comparison game for $\mathcal{M}, \mathcal{N}$,

(2) first-order sentences φ of quantifier depth k with $\mathcal{M} \models \varphi$ and $\mathcal{N} \not\models \varphi$.

A similar match exists for Duplicator, whose winning strategies correspond to well-known mathematical notions of similarity between models, in some cases: "isomorphism".

Determinacy, and further theoretical issues As long as we fix a finite duration k, Zermelo's Theorem still applies to model comparison games: either Duplicator or Spoiler must have a winning strategy. But actually, it is easy to imagine comparison games that go on forever: in that case, we say that Duplicator wins if she loses at no finite stage of such an infinite history. (This is a very natural feature game-theoretically, since infinitely repeated games are central in in "evolutionary games" modeling what happens in large communities of players over time.) It can be shown that infinite comparison games are still determined, since their structure is still quite simple. This abstract similarity from a game-theoretic perspective goes even further. It can be shown that comparison games are evaluation games at a deeper level, for logical languages that define structural similarities.

7.7 Bisimulation Games

Comparison games do not just apply to predicate logic. They are also widely used for the logics of information and action that you have seen in Chapters 5 and 6. Here is an important example, for the notion of bisimulation that was defined in Chapter 6 (Definition 6.44).

A straightforward modification of the above game is by restricted selection of objects, say, to relational successors of objects already matched. This leads to comparison games for epistemic and dynamic logic. The definition of bisimulation is repeated here, adapted to the present context of models $\mathcal{M}, \mathcal{N}$ for multi-modal logic.

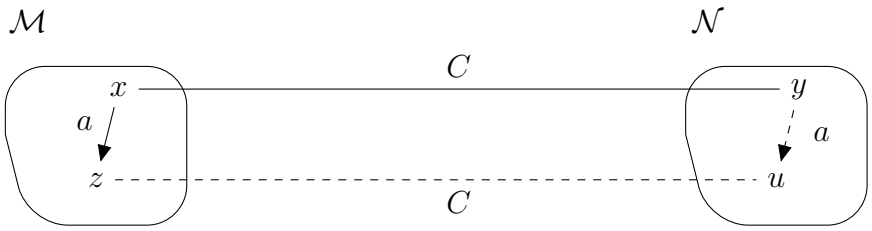
Definition 7.29 (Bisimulation) A bisimulation is a binary relation C between states of models $\mathcal{M}, \mathcal{N}$ with binary transition relations R_a, such that, whenever xCy, then we have 'atomic harmony' or 'invariance' (x, y satisfy the same proposition letters), plus two-way zigzag clauses for all relations a:

Invariance x and y verify the same proposition letters,

Zig If xR_az, then there exists u in $\mathcal{N}$ with yR_au and zCu.

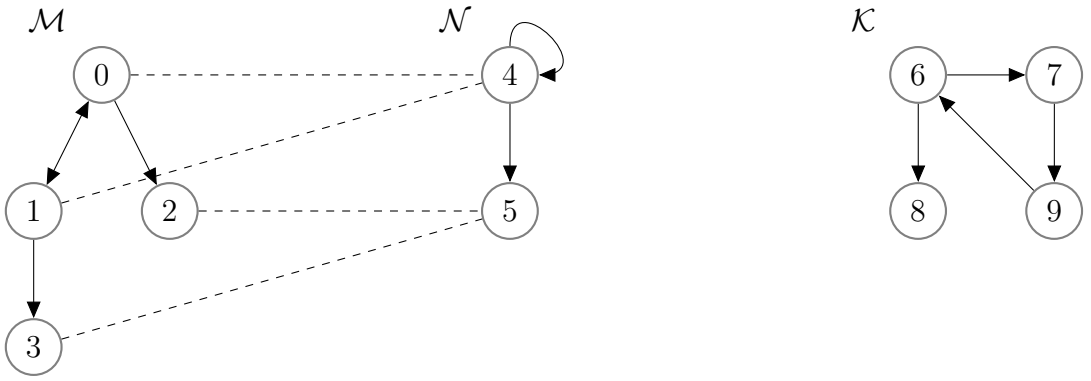
Zag vice versa.

The **Zig** condition in a picture:



This definition was already illustrated by some examples in Chapter 6. Here are some more examples.

Example 7.30 (Bisimulation between process graphs) State 0 in $\mathcal{M}$ and state 4 in $\mathcal{N}$ are connected by the bisimulation given by the dotted lines – but no bisimulation includes a match between world 4 in $\mathcal{N}$ and world 6 in $\mathcal{K}$:



We recall from Chapter 6 that modal formulas are invariant for bisimulation:

Proposition 7.31 (Invariance Lemma) If C is a bisimulation between two graphs $\mathcal{M}$ and $\mathcal{N}$, and mCn, then $\mathcal{M}, m$ and $\mathcal{N}, n$ satisfy the same modal formulas.

The fine-structure of bisimulation suggests a game comparing epistemic or dynamic models between Duplicator and Spoiler, comparing successive pairs (m, n) in two models $\mathcal{M}$, $\mathcal{N}$:

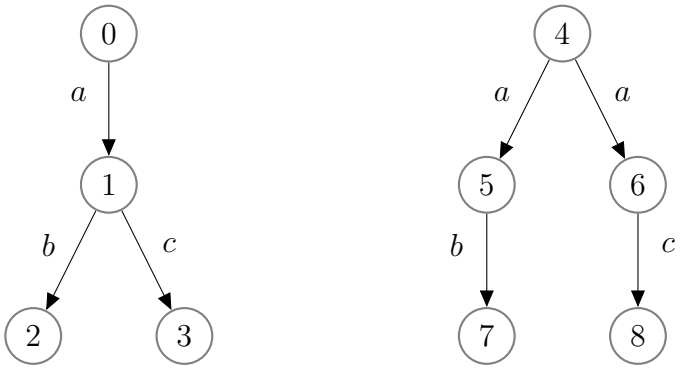
> In each round Spoiler chooses a state x in one model which is a successor of the current m or n, and Duplicator responds with a matching successor y in the other model. If x, y differ in their atomic properties, Spoiler wins. If Duplicator cannot find a matching successor: likewise.

Again one can show that this fits precisely:

(1) Spoiler's winning strategies in a k-round game between $(\mathcal{M}, s)$, $(\mathcal{N}, t)$ match the modal formulas of operator depth k on which s, t disagree.

(2) Duplicator's winning strategies over an infinite round game between $(\mathcal{M}, s)$, $(\mathcal{N}, t)$ match the bisimulations between them linking s to t.

Example 7.32 Spoiler can win the game between the following models from their roots. He needs two rounds – and different strategies do the job. One stays on the left, exploiting the difference formula $\Diamond_a(\Diamond_b\top \vee \Diamond_c\top)$ of depth 2, with three existential modalities. Another winning strategy switches models, but it needs a smaller formula $\Box_a\Diamond_b\top$.



In the non-bisimulation pair $\mathcal{N}$, $\mathcal{K}$ from above, repeated here, starting from a match between worlds 1 and 3, Spoiler needs three rounds to win.

Spoiler forces Duplicator in two rounds into a match where one world has no successor, while the other does. One winning strategy for this exploits the modal difference formula $\Diamond\Diamond\Box\bot$.

Exercise 7.33 Give a winning strategy for Spoiler in the game about the two process graphs in Exercise 6.22 from Chapter 6.

7.8 Preference, Equilibrium, and Backward Induction

Now we turn to real game theory. The games that we considered so far are trees of nodes (the stages of the game) and moves, that is, labeled transition relations. Moreover, the endpoints of the game were marked for winning or losing by the relevant players. Compare the trees for the game of Noughts and Crosses in Example 7.11. But this is not enough for real games. What is typical there is that players may have finer *preferences* between outcomes, that can lead to much better predictions of what rational players would achieve.

Definition 7.34 (Extensive Games with Perfect Information) An *extensive game with perfect information* consists of

(1) a set N of players,

(2) a set H of (finite) sequences of successive actions by players closed under taking prefixes

(3) a function P mapping each non-terminal history (i.e., one having a proper extension in H) to the player whose turn it is,

(4) for each player $i \in N$ a preference relation $\geq_i$ on the set of terminal histories (histories having no proper extension in H).

Without the preference relation, one has an *extensive game form*.

Example 7.35 (Donation, Extensive Game)

(1) There are two players I and II.

(2) There are two actions, giving a donation to the other player (d) and failing to do so (n). Distinguishing between players, these become D, d, N, n (capital letters for the first player).

The rules of the game are as follows. Each of the two players is given 10 euros. Each player is informed that a donation of 5 euros to the other player will be doubled. Next, the players are asked in turn whether they want to make the donation or not.

Assuming I plays first, the terminal histories are Dd, Dn, Nn, Nd. The set H consists of the terminal histories plus all proper prefixes:

$$\{\lambda, D, N, Dd, Dn, Nn, Nd\}$$

where λ is the empty list. The turn function P is given by $P(\lambda) = \text{I}, P(D) = \text{II}, P(N) = \text{II}$.

(3) To see what the preferences for I are, note that receiving a donation without giving one is better than receiving a donation and giving one, which is in turn better than not receiving a donation and not giving one, while giving a donation while receiving nothing is worst of all. So we get:

$$Nd >_1 Dd >_1 Nn >_1 Dn$$

The preferences for II are:

$$Dn >_2 Dd >_2 Nn >_2 Nd.$$

Definition 7.36 (Preferences and Payoff Functions) A payoff function (or: utility function) for a player i is a function u_i from game outcomes (terminal histories) to integers. A payoff function u_i *represents* the preference ordering $\leq_i$ of player i if $p \leq_i q$ iff $u_i(p) \leq u_i(q)$, for all game outcomes p, q.

Example 7.37 For I's preferences in the Donation game, we need a utility function u_1 with

$$u_1(Nd) > u_1(Dd) > u_1(Nn) > u_1(Dn).$$

The most obvious candidate for this is the function that gives the payoff for I in euros:

$$u_1(Nd) = 20, u_1(Dd) = 15, u_1(Nn) = 10, u_1(Dn) = 5.$$

For II this gives:

$$u_2(Dn) = 20, u_2(Dd) = 15, u_2(Nn) = 10, u_2(Nd) = 5.$$

Combining these payoff functions, we get:

$$u(Nd) = (20, 5), u(Dd) = (15, 15), u(Nn) = (10, 10), u(Dn) = (5, 20).$$

But there are other possible candidates for this. Here is an example (one of many):

$$u_1(Nd) = 3, u_1(Dd) = 2, u_1(Nn) = 1, u_1(Dn) = 0.$$

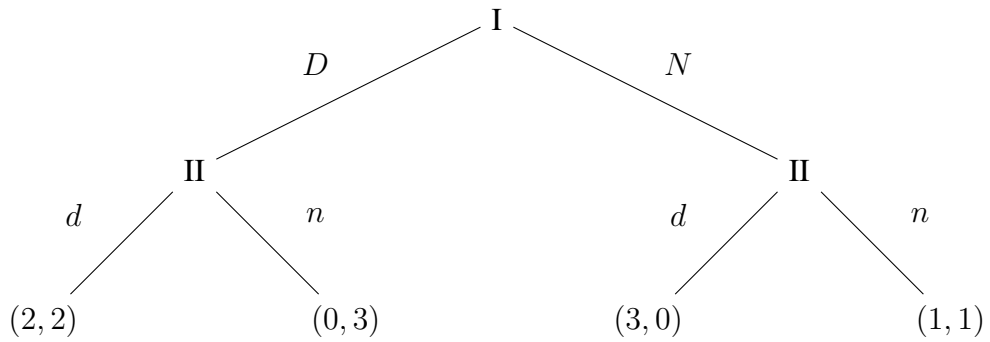
Similarly, for u_2, we can choose:

$$u_2(Dn) = 3, u_2(Dd) = 2, u_2(Nn) = 1, u_2(Nd) = 0.$$

Combining these payoff functions, we get:

$$u(Nd) = (3, 0), u(Dd) = (2, 2), u(Nn) = (1, 1), u(Dn) = (0, 3).$$

Such a combined payoff function can be used in the game tree for the Donation game, as follows:



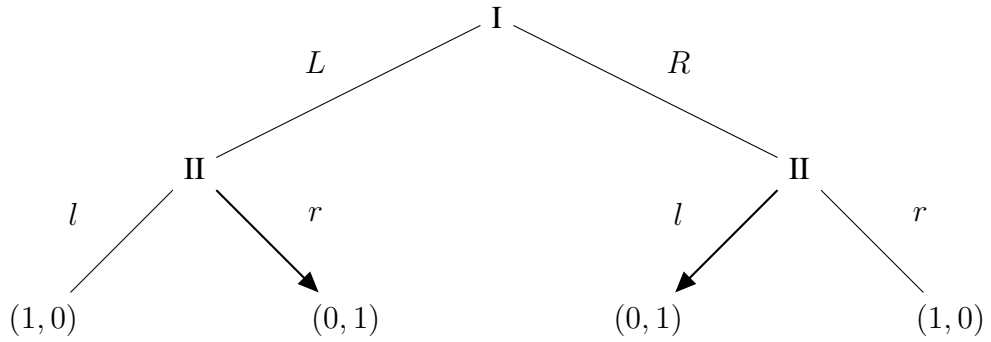
The definition of extensive games can easily be generalized to infinite games, where the action histories need not be finite. If we allow H to contain countably infinite histories, then we need to impose the extra condition of *closure under countable limits*. What this means is that if an infinite sequence of histories $h, h', \ldots$ is present in H where each history h_i extends the previous one, then the infinite history that has each h_i as a prefix

is also in H. We can generalize still further to infinite games of higher cardinalities. Mathematicians love to do this, not bothered at all by the fact that this breaks the link to real-world-games that people can play.

Pictorially, extensive games are finite (or infinite) mathematical trees, whose branches are the possible runs or histories of the game. You may already have recognized them as a structure that you have learnt about in Chapter 6: extensive game trees are obviously models for a dynamic logic with various sorts of labeled actions (the moves), and special atomic predicates given by a valuation (say, the marking for preference values of players, or indications whose players turn it is at some intermediate node). We will return to this "process perspective" on games later, since it is the starting point for their general logical analysis.

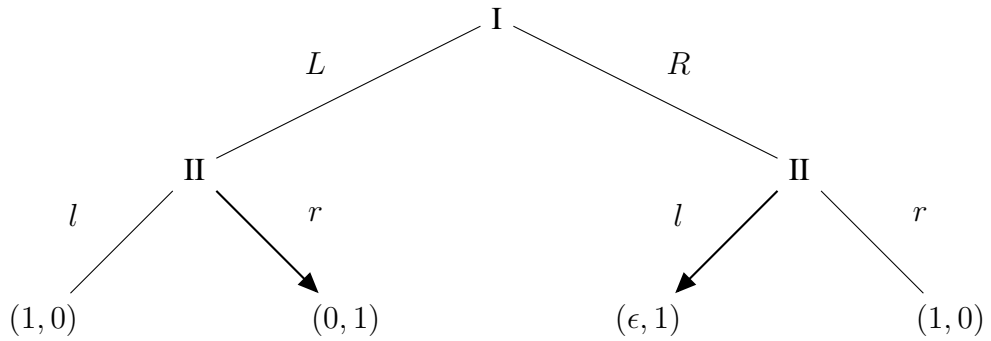
But for now, let us just look a bit closer at what preferences do. We start with a simple case where preferences may make a Zermelo-style analysis more interesting.

Example 7.38 (Losing with a little twist) Recall our very first example, where we now indicate players' evaluation of the outcomes in pairs (I-value, II-value):



A Zermelo computation tells us that II has a winning strategy indicated by the black arrows, and it does not matter what I does in equilibrium.

But now suppose that I has a slight preference between the two sites for his defeat, being the end nodes with values $(0, 1)$. Say, the one to the left takes place on a boring beach, where the sea will wash out all traces by tomorrow. But the one to the right is a picturesque mountain top, and bards might sings ballads about I's last stand for centuries. The new preferences might be indicated as follows:



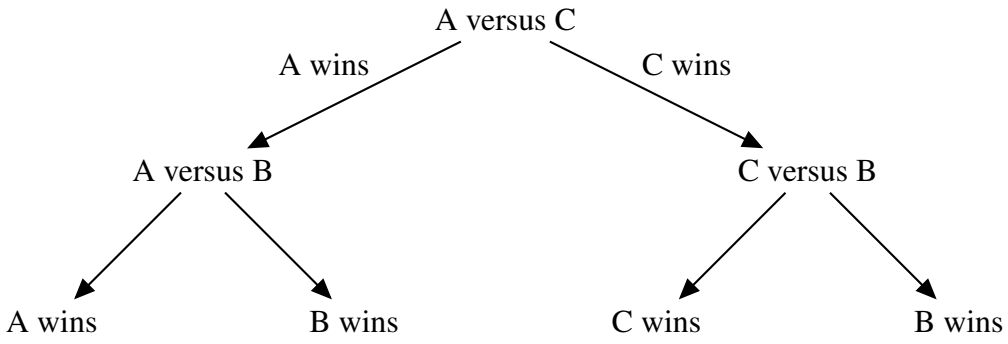
Intuitively, with these preferences, I goes 'right' at the start, and then II goes 'left'.

With preferences present, however, examples can quickly get highly non-trivial:

Example 7.39 (Tiered Voting) Three political parties 1, 2, 3 have the following preferences concerning issues A, B, C, indicated in order from top to bottom:

1	2	3
A	C	B
C	B	A
B	A	C

Moreover, the following schedule of voting has been agreed upon. First, there will be a majority vote between A and C, eliminating one of these. The winner will be paired against B. The game tree for this is as follows, where players move simultaneously casting a vote. We just record the outcomes, without their vote patterns:



Here is what will happen if everyone votes according to their true preferences. A will win against C, after which A will lose against B. But now 1 might reason as follows. "If I had voted for C against A in the first round (against my real preference), the last round would have been between B and C, which would have been won by C – which I prefer to outcome B." But other players can reason in the same way.

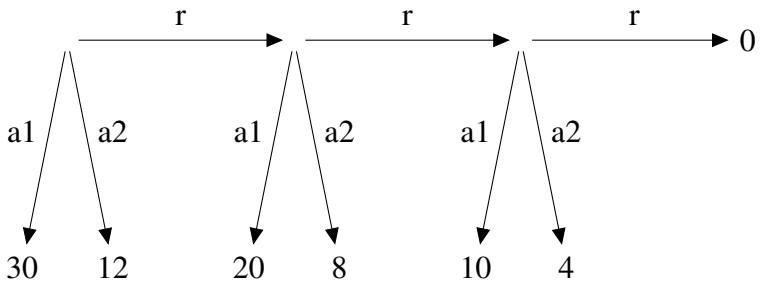
What is a stable outcome, representing rational behaviour of people in such a situation? Well, players cannot do better at pre-final nodes than state their true preferences. Any deviation will harm their favourite outcome. So players know the outcomes at the middle nodes of the procedure. Therefore, in the first round, players will vote according to their true preferences between those outcomes.

Backward Induction This brings us to the key notion of this section: **backward induction** is a general algorithm for computing a 'most rational course of action' by finding values for each node in the game tree for each player, representing the best outcome value she can guarantee through best possible further play (as far as within her power). Here is a formal definition.

Definition 7.40 (Backward Induction Algorithm) Suppose II is to move, and all values for daughter nodes are known. The II-value is the maximum of all the II-values on the daughters, the I-value the minimum of the I-values at all II-best daughters. The dual case for I's turns is completely analogous.

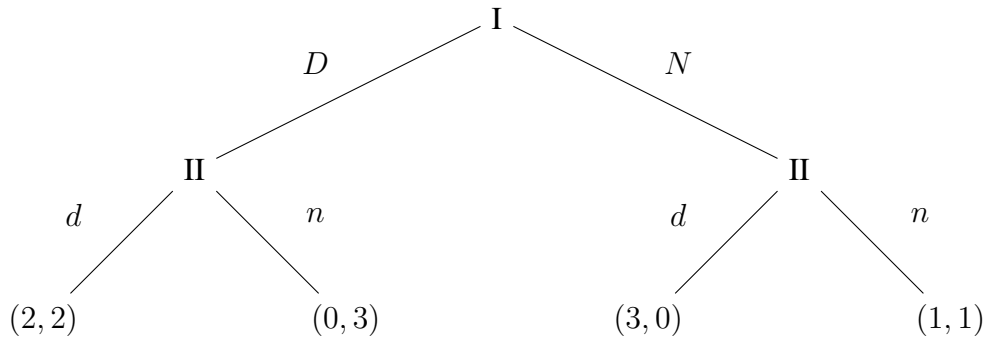
Backward induction is a useful tool for decision making.

Example 7.41 (Decision Making by Backward Induction) Consider a lady who decides that she is in need of a husband. She is a high flyer, with no time to spare for hanging around in bars and discos. So she agrees that a dating agency presents her with three candidates. She decides beforehand that she will marry one of these three, or no-one at all. She distinguishes three categories: a guy can be (1) great, (2) halfway decent, or (3) completely hopeless. She has confidence that the agency will be able to come up with category (1) and (2) candidates, and she estimates that the two kinds are equally likely. She puts the value of being married to a great guy at 10 per year, and the value of being married to a halfway decent guy (snores, drinks too much, but still decent enough to put the garbage out on Mondays) at 4 per year. The value of being single is 0. Taking a time horizon of 3 years, and given that the beginning of every year has one candidate in store for her, what should she do? Clearly, given her utilities, she should grab the first great guy that comes along. But suppose the agency offers her only a halfway decent guy? Then what? This decision problem can be pictured as follows:



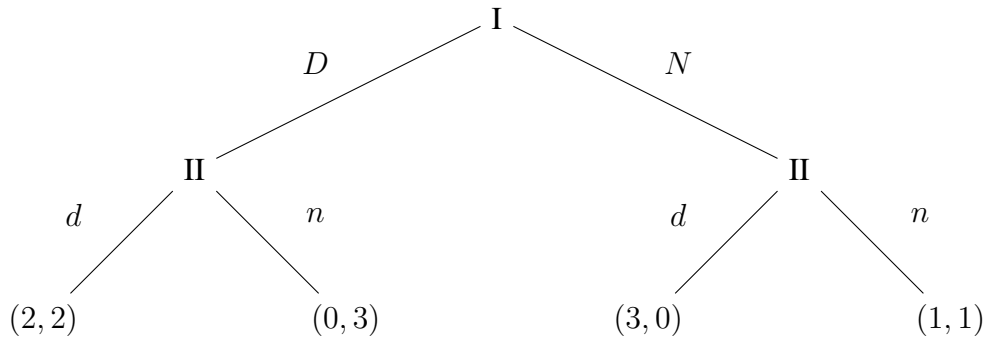
Reasoning backward, she first asks herself what she should do with the third candidate, supposing she has refused the earlier two. Clearly, given her utilities, she should accept him no matter what. Accepting a so-so guy would give her 4, and refusing him will give her 0. Now how about accepting a so-so guy as second candidate? Refusing him will give on average a payoff of 7, and accepting him gives her twice 4, which is 8. This is better, so it is rational for her to accept. But for the first candidate she can afford to be picky. Accepting a so-so guy would give her 12, and refusing him will give her on average 14, given that she accepts the second candidate no matter what, which she should, for the reasons we we have just seen.

Example 7.42 (BI Solution to the Donation game) Let's compute the 'rational course of action' in the Donation game (Example 7.35) by Backward Induction (BI):

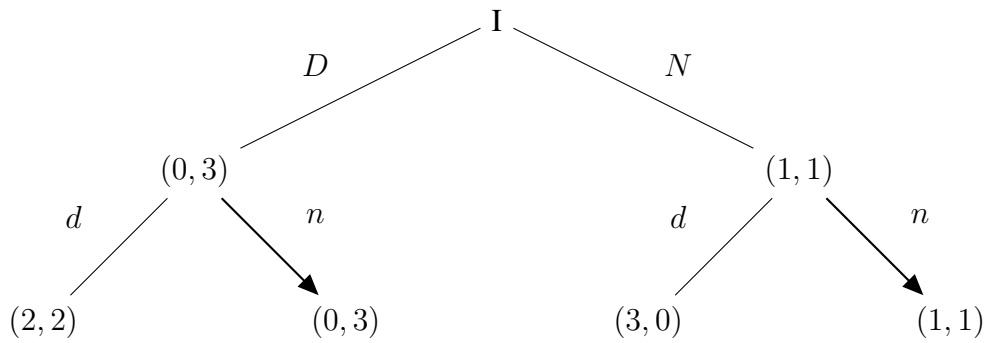


The II value for D is that of Dd, which is 2, the II value for N is that of Nn, which is 1. The I value for D is the I value of Dd, which is 2. The I value for N is the I value of Nn, which is 1. So BI dictates that I plays N, to which II will respond with n.

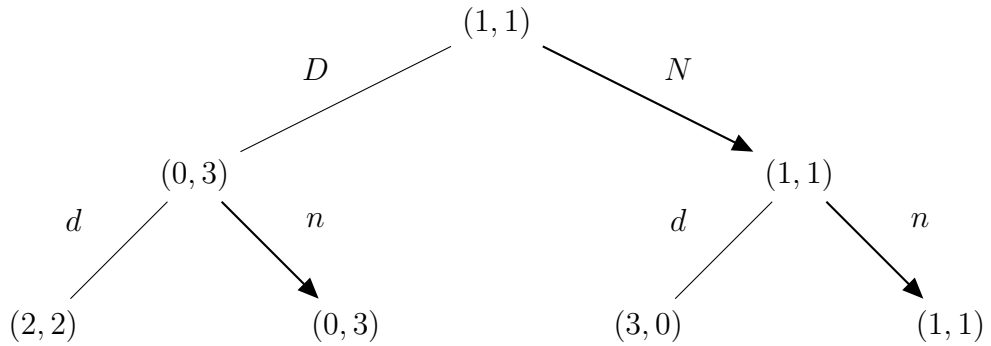
Backward Induction clearly generalizes the Zermelo algorithm that we have seen before. Instead of propagating win/lose information up the tree, we now propagate payoffs: from



to



and next to

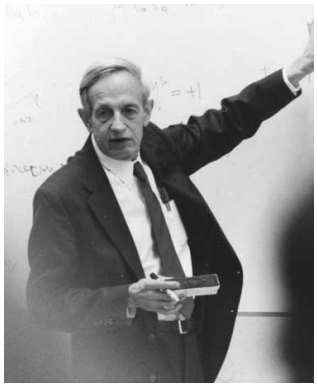


One can see it as a *maximin* procedure: players *maxi*mize their *min*imal gain. Again, its scope goes further than you might think. Algorithms computing numerical values like this also occur in AI, under the name '$\alpha\beta$ search'. In that case, the values at nodes indicate heuristic potentials for finding some desired goal.

Exercise 7.43 Using Backward Induction, compute how the parties should vote in the scenario of Example 7.39.

Strategies and equilibrium behaviour If you look at arrows moving from nodes to subsequent nodes where the current player gets her maximum value, Backward Induction computes a pair of *strategies* in our earlier sense, one for each player. Why do game theorists think that Backward Induction computes a "best" or "rational" behaviour in this manner? This has to do with the fact that these strategies are in *equilibrium*: no player has an incentive to deviate. To define this precisely, first note that any strategies σ, τ for two players determines a unique outcome $[\sigma, \tau]$ of the game, obtained by playing the two

strategies against each other.

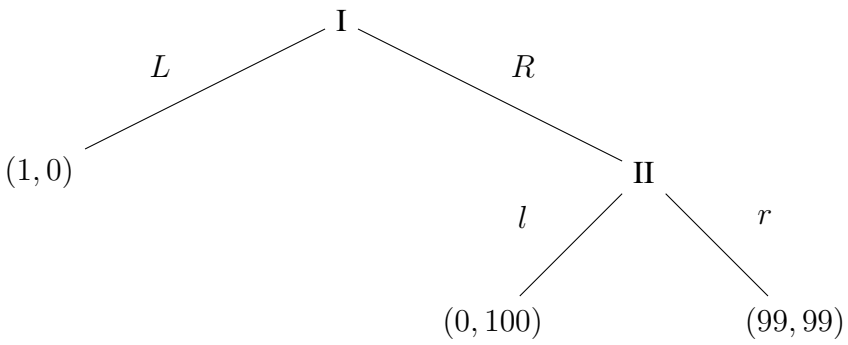


John Nash

Definition 7.44 (Nash equilibrium) A pair of strategies σ, τ in a two-player game is a *Nash equilibrium* if, for no $\sigma' \neq \sigma$, $[\sigma', \tau] \geq_1 [\sigma, \tau]$, and similarly for player II with respect to τ: for no $\tau' \neq \tau$, $[\sigma, \tau'] \geq_2 [\sigma, \tau]$. Here $[\sigma, \tau]$ signifies the outcome of the game when I plays σ and II plays τ. In other words, neither player can improve his outcome by deviating from his strategy while it is given that the other player sticks to hers.

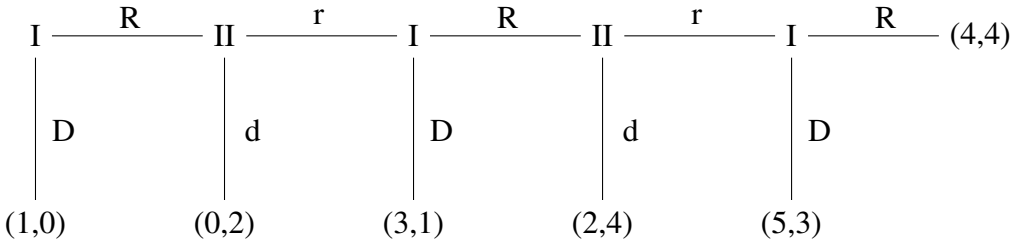
In our earlier logic games, any pair of a winning strategy plus any strategy for the other player is a Nash equilibrium. This shows that equilibria of a game need not be unique, and indeed, there can be one, more, or none at all. Backward Induction at least produces equilibria of a very special kind: they are "subgame-perfect". What this says is that the computed best strategies at nodes remain best when restricted to lower nodes heading subgames underneath. This property is not guaranteed by Nash equilibrium per se: think again of my playing badly in a logic game against an opponent playing a winning strategy. This is not perfect in subgames at lower nodes that are not reached, where I could have won after all by playing better.

Criticism Despite its appealing features, Backward Induction has also been criticized for being at odds with intuition in some cases. In the following simple game, Backward Induction computes an equilibrium with outcome $(1, 0)$, making both players hugely worse off than the outcome $(99, 99)$ that is also a possible outcome of the game:



This has been a starting point for analyzing the reasoning underlying this "obvious" algorithm in much more detail, and game theorists have found it useful to employ techniques from logic for this purpose. We will return to this issue later on. The following example is another case where Backward Induction yields an unintuitive result.

Example 7.45 (Centipede Games) A centipede game is a game where two players take turns and where each player can decide to opt out or play on. Opting out gives a better payoff than the opponent, but playing on raises the stakes: the sum of the payoffs increases. Here is an example.



Analyzing this, one sees that the players together would be best off by staying in the game until the very end, for then they will each receive 4. But will player I play R in his last move? Clearly not, for playing D will give a better payoff of 5 rather than 4. So player II realizes that staying in the game at the pre-final stage, will yield payoff 3. So opting out at this stage is better for her, so she plays d. Player I is aware of this, and to avoid this outcome, will play D, with outcome $(3, 1)$. So II realizes that this will be the result of playing r. She will therefore play d, with result $(0, 2)$. But this is bad for player I, who will therefore play D on his very first move, and the game ends with outcome $(1, 0)$.

Please note that we have now left the "game of logic" here, that is, the games of winning and losing that we used for logical tasks. We will now take a look at "logic of games": what is there for a logically trained mind to see in them?

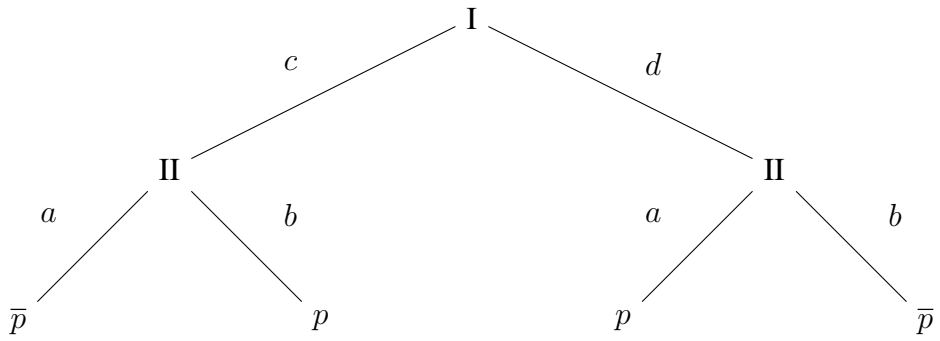
7.9 Game logics

We have seen that not all is well in finding game 'solutions' by the usual methods. To remedy that, we need to analyze the reasoning of the players more carefully. For that we introduce **game logics**.

Logics of game structure and strategies For a start, extensive games are processes in the same sense as Chapter 6. This means that everything you have learnt there applies at once.

Describing moves What would a logic for describing moves look like?

Example 7.46 (An extensive game tree) . Consider a game tree for two players I, II with four possible actions c, d, a, b, and some special property p holding at two of the four possible end states:



Here is a typical dynamic formula which is true at the root of his model:

$$[c \cup d]\langle a \cup b\rangle p.$$

Each of the actions c and d leads to a state where either a or b can be executed to get to a final state where p holds. In our earlier terms, this says that player II has a strategy ensuring that the outcome of the game satisfies p. Here, p might just be the property that II wins, in which case the modal formula expresses that II has a winning strategy.

Describing strategies The preceding style of description does not yet define the strategies themselves. But that, too, can be done with the techniques of Chapter 6, namely programs viewed as defining sets of transitions. The total move relation of a game is clearly a union of atomic transitions, and strategies are subrelations of the move relation, namely, transition functions defined on players' turns. (Arbitrary subrelations would be more like more loosely specified "plans".) Thus, on top of the 'hard-wired' moves in a

game, complex PDL-style relations can define strategies in terms of players options at a current node (*IF THEN ELSE*), sequential composition, and even iteration (as in a rule "always repeat the previous move by the other player until you have won").

Example 7.47 (Broken Record) As an example, here is a PDL version of the well-known 'broken record' strategy: whatever player I says (does), player II keeps repeating her message (action) b until I gives up:

$$(\text{move}_1; b)^*; ?\text{win}_2.$$

Example 7.48 (Match Removal Game) This is played between I and II. The player that is next to move removes 1, 2 or 3 matches from a pile. The player that can take the last match(es) has won. If the number of matches on the table is a four-fold, and I is next to move, the following is a winning strategy for player II:

$$((\text{one}_1; \text{three}_2) \cup (\text{two}_1; \text{two}_2) \cup (\text{three}_1; \text{one}_2))^*; ?\text{stack-empty}.$$

Exercise 7.49 Consider a finite game tree. Using the language of propositional dynamic logic, define the following assertion about players powers in the game:

> σ is a strategy for player i forcing the game, against any play of the others, to pass only through states satisfying φ.

Describing preferences Game trees are not only models for a dynamic logic of moves and strategies, but also for players preferences. In this course, we have not told you how to reason about preferences, even though this is an upcoming topic in studies of agency, since our behaviour is clearly not just driven by pure information, but just as much by what we want and prefer. Evidently, games involve information, action and preferences all intertwined. Indeed, a solution procedure like the above Backward Induction really depends on mixing these notions in a very specific way, that game theorists call "rationality": players only choose moves whose outcome they consider best for them, given what they know and believe about the game and the other players.

It would take us too far in this course to analyze Backward Induction in full logical style, but here is one typical fact about it. Let us add an operator

$$\langle \leq_i \rangle \varphi$$

to our logical language with the following intended meaning:

> There is some outcome of the game that player i finds at least as good as the present stage where the formula φ is true.

Then the key fact about the Backward Induction strategy σ, viewed as a program in our dynamic logic, can be stated as follows in logical terms:

Fact 7.50 The backward induction solution of a finite game is the unique binary relation bi on the game tree satisfying the following modal preference-action law:

$$[\text{bi}^*](\text{end} \to \varphi) \to [\text{move}]\langle \text{bi}^* \rangle (\text{end} \wedge \langle \leq_i \rangle \varphi)$$

for all formulas φ.

This looks extremely intimidating. But you may find it a useful exercise in reading logical formulas to see that it essentially says the following:

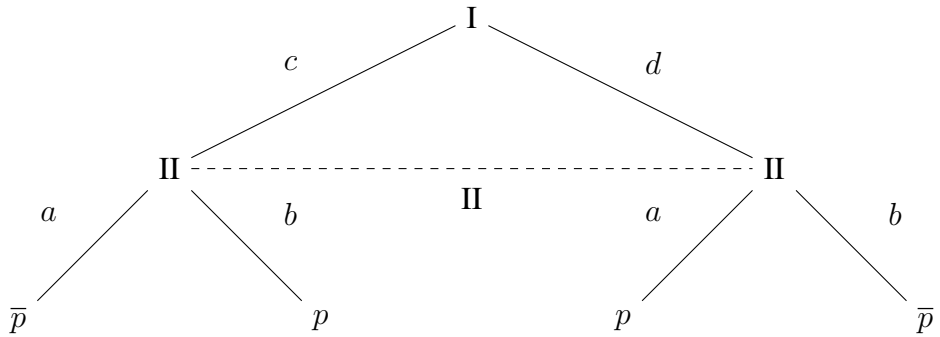
> There is no alternative move to the BI-prescription at the current node all of whose outcomes would be better than following the BI-solution.

7.10 Games with imperfect information

Logical analysis extends beyond the kinds of games that we have seen in this chapter so far. For instance, the ideas of Chapter 5 come into play with the extended class of games with imperfect information: that is, the players need not know exactly where they are in a game tree. This happens in many settings, for instance, when playing at cards where many things are not publicly known – and in this sense, our card examples of Chapter 5 were entirely appropriate.

Here we just show how logics of the sort you have studied apply to this broader setting.

Example 7.51 (An extensive game with imperfect information) Consider a game given earlier, in Example 7.46. But now assume we want to add an extra touch: player II is uncertain about the first move played by I. (Perhaps, I put it in an envelope, or perhaps this is a version of the donation game where there is no communication between the participants). This models a combined dynamic-epistemic language using ideas that you have seen in Chapters 5 and 6:



The modal formula $[c \cup d]\langle a \cup b \rangle p$ is still true at the root. But we can make more subtle assertions now, using the dotted line as an accessibility relation for knowledge. At stage

s, a player knows those propositions true throughout the 'information set' to which s belongs. Thus, after I plays move c in the root, in the left middle state, II knows that playing either a or b will give her p – the disjunction $\langle a\rangle p \vee \langle b\rangle p$ is true at both middle states:

$$\Box_2(\langle a\rangle p \vee \langle b\rangle p).$$

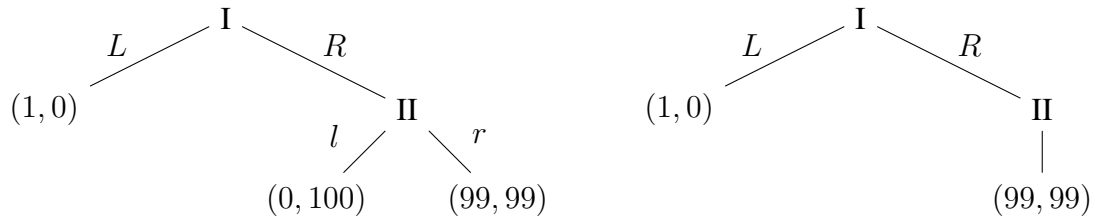
Nevertheless, there is no specific move of which II knows that it guarantees an outcome satisfying p – which shows in the leftmost middle state the truth of the formula

$$\neg\Box_2\langle a\rangle p \wedge \neg\Box_2\langle b\rangle p.$$

Think of a tragic person who knows the right partner is walking around right in this city, but does not know of any particular person whether (s)he is that partner.

Information dynamics Our final example is the information dynamics of Chapter 5, which again mixed information that agents have with changes in that information as events happen. Games typically have this dynamic flavour. As you play on, you learn more about what your opponent has done. But also, you can even change the whole game by exchanging information, as shown in the following scenario.

Example 7.52 (Making a promise) One can sometimes break a bad Backward Induction solution by changing the game. In our earlier game, the Nash equilibrium (1, 0) can be avoided by E's promise that she will not go left. This may be seen as a public announcement that some histories will not occur (E actually gives up some of her freedom) and the new equilibrium (99, 99) results, making both players better off:

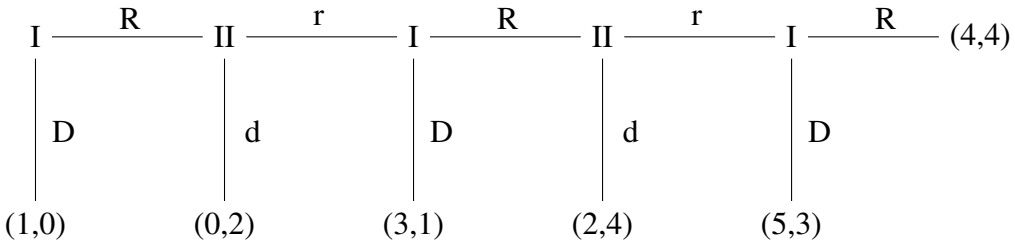


Another use of such dynamic actions is Backward Induction itself. We can view this procedure as a process of 'internal deliberation' via repeated announcements of 'rationality' that prunes the initial game tree:

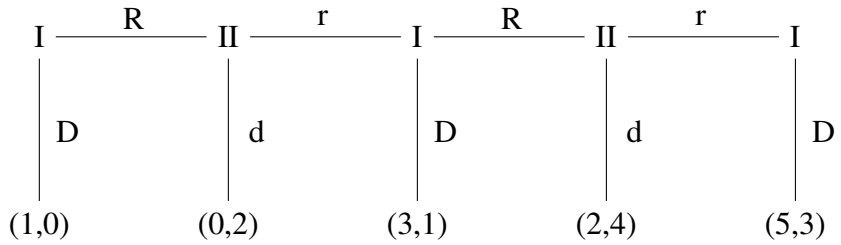
Theorem 7.53 The Backward Induction solution for extensive games is obtained through repeated announcement of the assertion "no player chooses a move all of whose further histories end worse than all histories after some other available move".

Instead of giving a proof, we show how the procedure works out for an example.

Example 7.54 (The Centipede Again) Consider the game from Example 7.45 again:



This has five turns, with I moving first and last. Stage 1 of the announcement procedure: I announces that he will not play R at the end. This rules out the branch leading to $(4, 4)$:



Next, stage 2. II announces that she will not play r. This rules out the state with payoff $(5, 3)$.

Stage 3: I announces that he will not play R. This rules out the state with payoff $(2, 4)$.

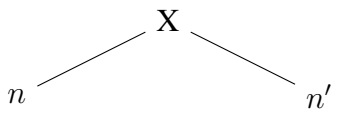
Stage 4: II announces that she will not play r. This rules out the state with payoff $(3, 1)$.

Stage 5: I announces that he will not play R. This rules out the state with payoff $(0, 2)$.

So I plays D and the game ends with payoff $(1, 0)$.

This scenario, in terms of repeated events of public announcements to the effect "I will act rationally, i.e., in my own best interest" removes nodes from the tree that are *strictly dominated* by siblings as long as this can be done.

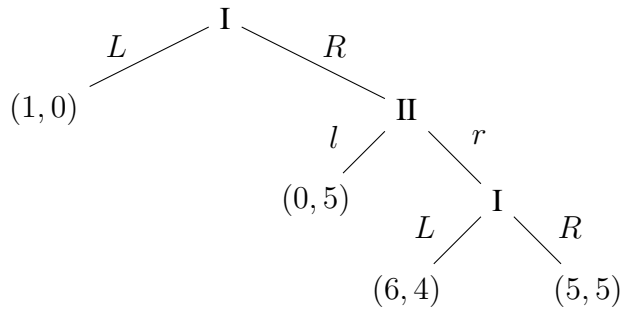
Definition 7.55 (Strict Domination) A node n in a game tree is strictly dominated by a sibling node n' if the player who is first to move (X in the picture) is better off by playing n' than playing n, no matter what the other players do.



Clearly, a rational player should never play a strictly dominated node. Technically, the iterated announcement procedure for extensive games ends in largest sub-models in which players have common knowledge of rationality in the above sense. This is one of the central notions in the foundations of game theory.

Other ways of reasoning about games We end with one more issue where logic meets the foundations of game theory today. Backward Induction is just one scenario for creating plausibility in a game. To see alternatives, consider what has been called a paradox in its reasoning. Assuming the above analysis, we expect a player to follow the BI path. So, if she does not, we must revise our beliefs about her reasoning. But then, why would we assume at all that she will play BI later on? BI seems to bite itself in the tail. Consider a concrete example:

Example 7.56 ('Irrational Behaviour' of Players) Backward Induction tells us that I will play L at the start in the following game:



So, if I plays R instead, what should II conclude? There are many different options, such as 'it was just an error, and I will go back to being rational', 'I is trying to tell me that he wants me to go right, and I will surely be rewarded for that', 'I is an automaton with a general rightward tendency', and so on.

Our logical analysis so far chooses for the interpretation that agents will always play rationally from the current stage onward. But this can be doubted, and in that case, logical analysis of games also needs an account of belief revision: the way in which we change our earlier beliefs about the game, and about the other players, as we proceed.

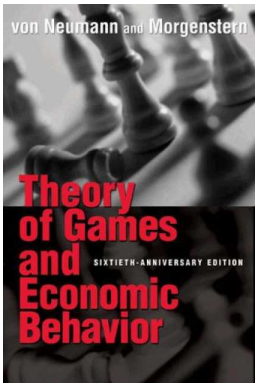
7.11 Logic and Game Theory

Game theory emerged in the course of the 20th century as the formal study of interactive decision making. Until recently, this field was perceived as rather far removed from logic. Key figures in its early development were John von Neumann and Oskar Morgenstern, who published their influential *Theory of games and economic behavior* in 1944, starting

off a new scientific discipline.

John von Neumann

Oskar Morgenstern



This section gives a brief introduction to the game theoretic way of thinking, in order to identify the many points of connection with logic.

First, here is a key definition that presents a different perspective on the extensive games we encountered before:

Definition 7.57 (Strategic Games) A *strategic game* consists of

(1) a finite set N of players,

(2) for each player $i \in N$ a non-empty set A_i of actions available to the player,

(3) for each player $i \in N$ a preference relation $\geq_i$ on $A = \prod_{j \in N} A_j$.

The members of $A = \prod_{j \in N} A_j$ are tuples $(a_1, \ldots, a_n)$, where a_1 is an action of the first player, a_2 an action of the second player, and so on. In a strategic game, there is no notion of temporal progression. The strategy of each player is viewed as condensed in a single action. So in a strategic game we consider the moves of all players simultaneously. The tuples in A are the possible global outcomes of the game, which can be evaluated by the players. The preference relation may also be encoded in terms of numerical utilities for players over outcomes, as explained before.

Many key notions and results in game theory work on strategic games, disregarding individual moves. This is the habitat of the matrixes for two-player games which most people probably associate with game theory.

Example 7.58 (Matching Pennies, Strategic Form) Players I and II both choose whether to show the head or tail of a coin. If the sides match, I gets 1 euro from II, if the sides are different, II gets 1 euro from I.

The matrix indicates all actions, possible outcomes, and their numerical utilities with that of I stated first:

	h	t
H	$1, -1$	$-1, 1$
T	$-1, 1$	$1, -1$

This game may be viewed as an analog of the one in 7.3. There, I (Falsifier) chose one object out of two, and then II (Verifier) chose one, with equality of the objects chosen being the criterion for winning or losing. In Matching Pennies, players choose simultaneously, or in ignorance of what the other did. This changes their powers considerably. E.g., unlike in 7.3, no one has a clear winning strategy here. The game has no Nash equilibrium in Ht, Hh, Th, Tt. The strategy pair Hh is not Nash, for Ht is better for II. Ht is not Nash, for Tt is better for I. Th is not Nash, for Hh is better for I, Tt is not Nash, for Th is better for II.

Still, we can ask ourselves what is the best one can do if one is forced to play the game of Matching Pennies repeatedly. Clearly, the answer to this is that randomly choosing between showing heads or showing tails, with equal probability, ensures that neither of the players will lose money. This motivates the following definition.

Definition 7.59 (Mixed Strategies) A mixed strategy for a player in a strategic game is a probability distribution over the player's possible actions.

Example 7.60 (Mixed Strategies for Matching Pennies) The mixed strategies $(\frac{1}{2}, \frac{1}{2})$ for both players in the Matching Pennies game form a Nash equilibrium: if one player plays this strategy, then deviation from the probability distribution $(\frac{1}{2}, \frac{1}{2})$ for II will make no difference for the outcome. For let $(p, 1-p)$ be a probability distribution for II, and assume $\frac{1}{2} < p \leq 1$. Then the good outcome Th for II will occur with probability $\frac{1}{2}p$, and the good outcome Ht with probability $\frac{1}{2}(1-p)$. The probability of a good outcome for II is $\frac{1}{2}p + \frac{1}{2}(1-p) = \frac{1}{2}$. In other words, as long as I plays $(\frac{1}{2}, \frac{1}{2})$, it makes no difference which mix II plays. This shows that $(\frac{1}{2}, \frac{1}{2})$ versus $(\frac{1}{2}, \frac{1}{2})$ is indeed a Nash equilibrium.

Exercise 7.61 Show that no other pair of mixed strategies is a Nash equilibrium for Matching Pennies. In particular, if one player plays a particular action with probability $p > \frac{1}{2}$, then the other player can exploit this by playing a pure strategy. But the resulting pair of strategies is not Nash.

Notice that the game of Matching Pennies is zero-sum: one player's gain is the other player's loss. This is not the case in the Donation game, or in the famous Prisoner's Dilemma.

The Dilemma of the Prisoners is probably the most famous example of game theory. For those who have never seen it, here is an informal description. Two players I and II are in prison, both accused of a serious crime. The prison authorities try to lure each of them into making a statement against the other. They are each promised a light sentence as a reward for getting their partner in crime convicted. If the prisoners both keep silent, they will get off with a light sentence because of lack of evidence. If one of them keeps silent but the other starts talking, the one who keeps silent is going to serve a considerable time

in prison and the other is set free. If both of them talk they will both get a medium term sentence.

Example 7.62 (Prisoner's Dilemma, Strategic Form) Here is the Prisoner's Dilemma in matrix form:

	s	b
S	$2, 2$	$0, 3$
B	$3, 0$	$1, 1$

Note that the payoff function is the same as in the Donation game (Example 7.35) The difference is that the Prisoner's Dilemma game is not played sequentially.

Why is this non-zero-sum game an evergreen of game theory? Because it is a top-level description of the plight of two people, or countries, who can either act trustfully or not, with the worst outcome that of being a sucker. For an armament race version, read the two options as 'arm' or 'disarm'.

The pair of strategies Bb is the only Nash equilibrium of the game: if the other one betrays me, there is nothing better I can do than also betray. For all other strategy pairs, one of the players is better off by changing his action.

In the Prisoner's Dilemma, the players have no rational incentive to coordinate their actions, and they end up in a situation that is worse than what would have resulted from their collaboration. This notion of being 'worse of' is made precise in the following definition.

Definition 7.63 (Pareto optimum) A *Pareto optimum* of a game is an outcome that cannot be improved without hurting at least one player.

Example 7.64 (The Prisoner's Dilemma Again)

	s	b
S	$2, 2$	$0, 3$
B	$3, 0$	$1, 1$

The Pareto optima are Ss, Sb, Bs. The Nash equilibrium Bb is *not* a Pareto optimum.

Example 7.65 (Tragedy of the Commons) This was made famous by Garrett Hardin in his classic essay, still available on internet, and very much recommended:

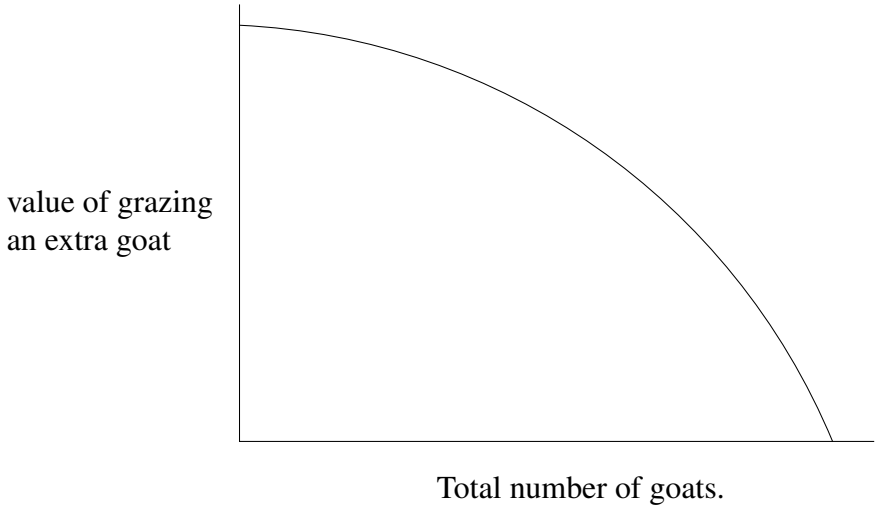
```
www.garretthardinsociety.org/articles/art_tragedy_of_
the_commons.html
```

Essentially, the Tragedy of the Commons is a multi-agent version of the Prisoner's Dilemma.

> The tragedy of the commons develops in this way. Picture a pasture open to all. It is to be expected that each herdsman will try to keep as many cattle as possible on the commons. Such an arrangement may work reasonably satisfactorily for centuries because tribal wars, poaching, and disease keep the numbers of both man and beast well below the carrying capacity of the land. Finally, however, comes the day of reckoning, that is, the day when the long-desired goal of social stability becomes a reality. At this point, the inherent logic of the commons remorselessly generates tragedy. [Har68]

Bringing more and more goats to the pasture will in the end destroy the commodity for all. Still, from the perspective of an individual herdsman it is profitable until almost the very end to bring an extra goat.

The following picture illustrates the dilemma:



Or view this as a game of an individual herdsman II against the collective I. Then the matrix is:

	m	g
M	$2, 2$	$0, 3$
G	$3, 0$	$-1, -1$

Each player has a choice between g (adding goats) and m (being moderate). Assuming that the collective is well-behaved, it pays off to be a free rider. But if everyone acts like this, system breakdown will result.

Of course, the general import of the matrices of strategic games is not the particular story *per se*, but rather their standing proxy for frequent types of social situation.

Example 7.66 ('Tragedy of the Commons' Scenario) The Tragedy of the Commons game describes a general mechanism that is rational for the one and disastrous for the many. Such mechanisms abound in the world around us:

- Citizens of Amsterdam who want cheap parking in the inner city;
- prosperous families wanting to drive bigger and bigger SUVs;
- airport hubs wanting to attract ever more air traffic;
- fishermen roaming the oceans in ever bigger fishing trawlers;
- logging companies cutting down ever more tropical forest;
- developed countries exporting their industrial waste to developing countries;
- US citizens defending the Second Amendment right to keep and bear firearms ("NRA: The largest civil-rights group ever").

It should be noted that slight differences in payoff function result in strikingly different scenarios.

Example 7.67 (Hawk versus Dove) Being aggressive against someone who is passive is advantageous. Being passive against someone who is also passive is so-so. Being aggressive against an aggressor can be disastrous. This gives the following matrix for the 'Hawk' versus 'Dove' game, where two players have the choice between aggressive and meek behaviour:

	h	d
H	$0, 0$	$1, 3$
D	$3, 1$	$2, 2$

This example also occurs frequently in biology. What is the best behaviour for two people or animals in a single encounter? And in the long run, what will be stable populations of predators playing Hawk and prey playing Dove?

'Hawk versus Dove' has two Nash equilibria, viz. Hd and Dh. In neither situation can anyone better himself by unilaterally switching strategies, while in the other two, both players can.

Exercise 7.68 What are pure strategy Nash equilibria for Hawk versus Dove? (Note: 'pure' means that actions are either played with probability 1 or with probability 0.)

Example 7.69 (Vos Savant's Library Game) The following story is from a column by Marilyn Vos Savant, San Francisco Chronicle, March 2002.

> A stranger walks up to you in the library and offers to play a game. You both show heads or tails. If both show heads, she pays you 1 dollar, if both tails, then she pays 3 dollars, while you must pay her 2 dollars in the two other cases. Is this game fair?

Let's put this in matrix form, with the stranger as the row player:

	h	t
H	$-1, 1$	$2, -2$
T	$2, -2$	$-3, 3$

You may think it is fair, for you can reason that your expected value equals

$$\frac{1}{4} \cdot (+1) + \frac{1}{4} \cdot (+3) + \frac{1}{2} \cdot (-2) = 0.$$

Vos Savant said the game was unfair to you with repeated play. The stranger can then play heads two-thirds of the time, which would give you an average pay-off of

$$\frac{2}{3}(\frac{1}{2} \cdot (+1) + \frac{1}{2} \cdot (-2)) + \frac{1}{3}(\frac{1}{2} \cdot (+3) + \frac{1}{2} \cdot (-2)) = -\frac{1}{6}.$$

But what if I play a different counter-strategy against this, viz. "Heads all the time"? Then my expected value would be

$$\frac{2}{3} \cdot (+1) + \frac{1}{3} \cdot (-2) = 0.$$

So, what is the fair value of this game – and should you engage in it? We will take this up again in Example 7.73 below.

Example 7.70 (Making sense) Linguistic expressions may be ambiguous, referring to more than one situation. This helps keep code short in communication, whereas unambiguous expressions tend to be elaborate and costly to process. Let A have two meanings: it can refer to situation X or Y. B is unambiguous, referring only to X, and C only to Y. The complexity of B, C is greater than that of A, in some intuitive sense. A speaker strategy is a choice of expression for each of the situations X, Y, while a hearer strategy decodes expressions into situations. Here are the possible strategies for both, in matrix form:

Speaker:

	X	Y
S_1	A	C
S_2	A	A
S_3	B	A
S_1	B	C

Hearer:

	A	B	C
H_1	X	X	Y
H_2	Y	X	Y

Let there be a known chance that situation X obtains versus Y: say $\frac{2}{3}$. First, Speaker says something, then Hearer interprets it. As for players' utilities, both prefer correct decodings to incorrect ones, and given that, less complex expressions to more complex ones. Linguistic behaviour amounts to pairs of strategies (S_i, H_j). This setting is called a *signalling game*. Is this enough to predict the observed behaviour of language users,

which is that the ambiguous expression is used for the most frequent situation, whereas the less frequent situation is referred to by its unambiguous code?

'Making Sense' has two Nash equilibria, viz. (S_1, H_1) and (S_3, H_2). The first of these represents the intended outcome. The second describes a situation where the ambiguous expression is used for the less frequent situation.

The notion of a Nash equilibrium remains the same in the larger strategy space where mixed strategies are allowed. Of course, outcomes will now be computed as expected values in the obvious sense. E.g., as we have seen, to do the best they can in Matching Pennies, players should play each action 'Heads', 'Tails' with probability 0.5. This will guarantee an optimal expected value 0 for both. Here is perhaps the most celebrated result from game theory.

Theorem 7.71 (von Neuman, Nash) All finite strategic games have equilibria in mixed strategies.

Rather than give a proof, we will illustrate this for the case of games with 2×2 matrices. For a strategy pair σ, τ in equilibrium yielding value $[\sigma, \tau]$, we call ϵ a best response for I to τ if $[\epsilon, \tau] = [\sigma, \tau]$. In other words, playing ϵ rather than σ against τ does not change the payoff. Now here is a useful fact.

Fact 7.72 If the strategy pair σ, τ is in equilibrium, then each pure strategy occurring in the mixed strategy σ is also a best response for player I to τ.

Proof. If some component pure strategy S gave a lower outcome against τ then we could improve the outcome of σ itself by decreasing its probability of playing S. □

We can use this to analyze Vos Savant's library game.

Example 7.73 (Library Game, Continued from Example 7.69) In equilibrium, suppose the stranger plays Heads with probability p and Tails with $1 - p$. You play heads with probability q and Tails with probability $1 - q$. By Fact 7.72, your expected outcome against the p-strategy should be the same whether you play Heads all the time, or Tails all the time. Therefore, the following equality should hold:

$$p \cdot 1 + (1 - p) \cdot (-2) = p \cdot (-2) + (1 - p) \cdot 3.$$

Working this out yields: $p = \frac{5}{8}$. By a similar computation, q equals $\frac{5}{8}$ as well. The expected value for you is

$$\frac{5}{8} \cdot \frac{5}{8} \cdot 1 + \frac{5}{8} \cdot \frac{3}{8} \cdot (-2) + \frac{5}{8} \cdot \frac{3}{8} \cdot (-2) + \frac{3}{8} \cdot \frac{3}{8} \cdot (3) = -\frac{1}{8}.$$

Thus, the game is indeed unfavourable to you – though not for exactly the reason given by Vos Savant.

Note that probabilistic solutions, for games like Matching Pennies or the Library Game, make most sense when we think of repeated games where you can switch between Heads and Tails.

But there are also other interpretations of what it means to play a mixed strategy. For instance, by a similar computation, besides its two equilibria in pure strategies, the Hawk versus Dove game has an equilibrium with each player choosing Hawk and Dove 50% of the time. This can be interpreted biologically in terms of stable populations having this mixture of types of individual.

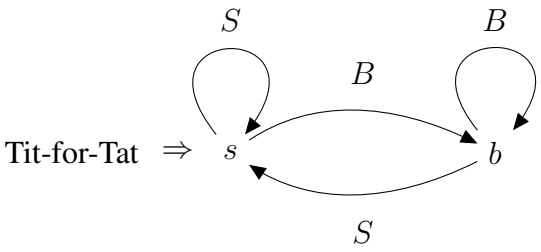
Conclusion As we said at the beginning of this chapter, our aim has not been to develop one more standard logical system. Instead, we have shown how logic and games are a natural meeting ground, where the themes of earlier chapters all return. We showed how predicate logic can be analyzed using special "logic games" of evaluation and model comparison. But we also showed how our logics of information and action apply to games in the general sense of game theory. These two directions are related: logics can be used to analyze games, but conversely games can also be used to analyze logics. This intriguing duality is far from being completely understood – but at least, you now know what it is about.

7.12 Outlook — Iterated Game Playing

Infinite games One striking trend in modern game theory is the evolutionary theory of infinitely repeated games. One source for this is the Prisoner's Dilemma game. This has only one Nash equilibrium, in which both players choose 'betray', even though both keeping silent would make both players better-off. Many amendments have been proposed since the problem was first proposed in the 1950s. In particular, it has become clear that one needs to look at repetitions of games like this, allowing for reactions to observed behaviour in the past. For then, we can punish or reward our opponents' previous behaviour. Now, fixed finite repetitions of games like Prisoner's Dilemma are of no help. A backward induction argument shows that, working back from the final play where retaliation is impossible, the 'bad' equilibrium Bb comes out best after all. But with infinite repetitions, and some natural 'discounting' of utilities of games further in the future, new equilibria emerge. An example made famous by Axelrod [Axe84] is:

> **Tit-for-Tat**: Copy one's opponents last choice, thereby giving immediate, and rancour-free, rewards and punishments.

Here is a process picture of this strategy (for player II, against player I):



As long as I sticks to S, respond with s, as soon as I plays B, respond with b, keep playing b as long as I plays B, and as soon as I plays S again, be forgiving and switch back to s.

It can be shown that (Tit-for-Tat,Tit-for-Tat) is a Nash equilibrium in the infinite Prisoner's Dilemma. Hence, cooperation is at least a stable option in the long run. The backdrop for this result are the 'folk theorems' of game theory showing that repeated versions of a game have a huge strategy space with many new equilibria. There is also a flourishing literature on showing when such a cooperative equilibrium will emerge in a population. One relevant line of research here is the learning theory of infinite games, where certain equilibria are learnt under plausible assumptions.

A complete analysis of infinite games in this sense requires the mathematics of dynamical systems with special leads from biology for setting up plausible systems equations. Such considerations over time are very rare in logic, at least so far. Sometimes, though, these considerations can be pushed back to simple scenarios that also make sense in logical analysis. Here is a nice illustration that makes sense when thinking about the stability of rules, say of some logical or linguistic practice.

Example 7.74 (Mutant Invasion) Consider a population playing some strategy S in an infinitely repeated game of encounters between 2 agents, with (S, S) a Nash equilibrium. E.g., S could be some logico-linguistic convention, like 'speaking the truth'. Now suppose that a small group of mutants enters, playing strategy F in every encounter. Let the probability that a mutant encounters another mutant be ϵ, typically a small number. Then the expected utility of any encounter for a mutant can be computed as follows:

$$\epsilon \cdot \text{utility}_M(F, F) + (1 - \epsilon) \cdot \text{utility}_M(F, S) \qquad \text{(mutant value)}$$

For members of the original population, the expectation lies symmetrically:

$$\epsilon \cdot \text{utility}_P(S, F) + (1 - \epsilon) \cdot \text{utility}_P(S, S) \qquad \text{(normal value)}$$

Here is an attractive notion of biology describing stability of a situation. A population is 'evolutionarily stable' if mutant invasions fizzle out. That is,

$$\text{for every strategy } F \neq S, \text{mutant value} < \text{normal value}.$$

By a simple calculation, this condition can be simplified, at least for 'symmetric' games where

$$\text{utility}_M(F, S) = \text{utility}_P(S, F).$$

It then becomes this qualitative notion.

Definition 7.75 A strategy S in a game is evolutionarily stable if we have both

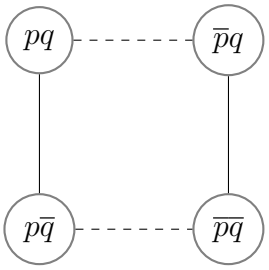
(1) (S, S) is a Nash equilibrium, and

(2) for every different best response S' to S, utility$(S', S') <$ utility(S, S').

This notion has additional bite. E.g., Tit-for-Tat, though a Nash equilibrium against itself, is not evolutionarily stable in the repeated Prisoner's Dilemma. Of course, other scenarios are possible. E.g., if mutants stick together, increasing their chances of mutual encounters, the above computations fail, and invasion may be possible after all.

7.13 Outlook — Knowledge Games

Information exchange in the sense of Chapter 5 can also be viewed as a game. We give a brief sketch. Players I and II both have a secret: I secretly knows about p and II secretly knows about q. Here is a picture of the situation (solid lines for I accessibilities, dashed lines for II accessibilities):



Both players would like to learn the other's secret, but are reluctant to tell their own. They don't want to tell their own secret without learning the secret of the other. They both have a choice: telling their own secret or not. The choice for I is that between $\pm p$ (telling whether p is the case) and $\top$ (uttering a triviality). II can choose between $\pm q$ and $\top$. The preferences for I are:

$$\top\pm\, q >_1 \pm p \pm q >_1 \top\top >_1 \pm p\top.$$

The preferences for II are:

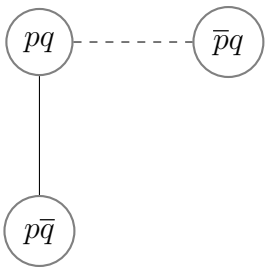
$$\pm p\top >_2 \pm p \pm q >_2 \top\top >_2 \top\pm\, q.$$

This situation is represented by the following strategic game:

	$\pm q$	$\top$
$\pm p$	$2, 2$	$0, 3$
$\top$	$3, 0$	$1, 1$

This game has one pure Nash equilibrium, $\top\top$.

Now consider a variation of this, where both players know that $p \vee q$. The model becomes:



Now more complex strategies make sense. Consider the following I-strategy: "If I knows that p then I keeps silent, otherwise I reveals $\neg p$". Formally:

$$(?\Box_1 p; !\top) \cup (?\neg\Box_1 p; !\neg p).$$

Exercise 7.76 Show that playing this strategy against $\top$ is an equilibrium.

7.14 Outlook — Games and Foundations

As we have seen, a game in which one of the two players has a winning strategy is called determined. Now, are all games determined? With this simple question, we are right in the foundations of set theory. Examples have been found of infinite non-determined games, but their construction turned out to depend strongly on the mathematical axioms one assumes for sets, in particular, the famous 'Axiom of Choice'. Therefore, in 1962 it has been proposed (by Jan Mycielski and Hugo Steinhaus) to turn the tables, and just *postulate* that all games are determined. This 'Axiom of Determinacy' states that all two-player games of length ω with perfect information are determined.

Since the Axiom of Choice allows the construction of non-determined two-player games of length ω, the two axioms are incompatible. From a logical point of view, the Axiom of Determinacy has a certain appeal. We have in finitary logic

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 \varphi(x_1, x_2, x_3, x_4) \vee \exists x_1 \forall x_2 \exists x_3 \forall x_4 \neg\varphi(x_1, x_2, x_3, x_4),$$

and so on, for longer $\forall\exists$ alternations of quantifiers. The infinitary version of this runs:

$$\forall G \subseteq S^\infty :$$

$$\forall x_1 \in S \exists x_2 \in S \forall y_1 \in S \exists y_2 \in S \forall z_1 \in S \exists z_2 \in S \ldots : (x_1, x_2, y_1, y_2, z_1, z_2, \ldots) \in G$$

iff

$$\exists x_1 \in S \forall x_2 \in S \exists y_1 \in S \forall y_2 \in S \exists z_1 \in S \forall z_2 \in S \ldots : (x_1, x_2, y_1, y_2, z_1, z_2, \ldots) \notin G.$$

But this *is* a formulation of the Axiom of Determinacy. Indeed, the Axiom of Determinacy might be viewed as 'Excluded Middle run wild', but then a gallop with beautiful mathematical consequences. There is a broad consensus today that set theory needs new axioms, but much less: which ones, and Determinacy is just one option. In any case, it may be said that games are an important source of intuitions here.

7.15 Outlook — Games, Logic and Cognition

The arena of game theory has many intriguing examples where there is a mismatch between what game theoretical considerations would predict and what actually happens if the games are played. Here is one much-discussed example:

Example 7.77 (The Ultimatum Game) Player I is shown a substantial amount of money, say 1000 euros. He is asked to propose a split of the money between himself and player II. If player II accepts the deal, they may both keep their shares, otherwise they both receive nothing. If this game is played once, a split $(999, 1)$ should be acceptable for II. After all, receiving 1 euro is better than receiving nothing. But this is not what we observe when this game is played. What we see is that II rejects the deal, often with great indignation.

Considerations about repeated play, reputation mechanisms, psychological factors, have been called to the rescue to explain what happens.

Other examples where what we observe in reality seems at odds with game theoretical rationality are the centipede games, discussed above in Examples 7.45 and 7.54. Instead of the first player immediately opting out of the game, players often show partial cooperation. Maybe they reason that it is better to cooperate for a while, and then defect later, when there is a better reward for the evil deed? It has also been suggested that this has something to do with limitations in our cognitive processing. We all can do 'first order theory of mind': imagine how other people think about reality. Some of us do 'second order theory of mind': imagine how other people think about how *we* think about reality. Very few people take the trouble to move to higher orders. But in a backward induction argument for a centipede game this is what seems to be going on, and on and on ...

Summary of Things You Have Learnt in This Chapter *You have become aware of the natural fit between games and logic, in a number of areas. You have learnt to see reasoning about logical consequence (argumentation) as a game. You know how to play an evaluation game. You know the concept of a winning strategy, and you understand Zermelo's theorem and the algorithm behind it. You have learnt how to apply Zermelo's procedure to find winning strategies for finite zero-sum two-player games such as Sabotage. You know how to play model comparison games, and you know what a difference formula is. You are able to find winning strategies in bisimulation games. You understand*

the concept of a Nash equilibrium, and you are able to solve games by backward induction. Finally, you understand the basic game-theoretic notions of a strategic game and of a mixed strategy solution, and you understand how well-known strategic games like the Prisoner's Dilemma and Hawk versus Dove stand proxy for social situations.

Further Reading A recent textbook that explores and explains the connections between games and logic is [VÏ1]. An illuminating paper on the importance of the game-theoretic perspective in logic is [Fag97].

The book that started game theory, [NM44], was already mentioned above. There are many excellent textbooks on game theory: [Str93] and [Osb04] are among our favourites. A more light-hearted introduction is [Bin92].

The use of game theory to investigate the games we play when conversing with each other in natural language is demonstrated in [Cla12].

Methods

Chapter 8

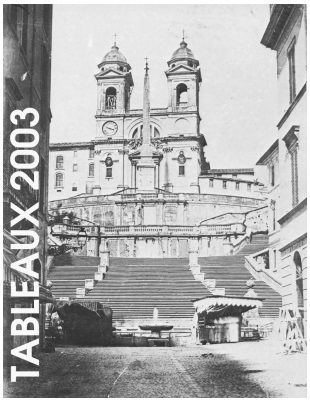
Validity Testing

In the first three chapters various methods have been introduced to decide the validity of different sorts of inferences. We have discussed truth-tables and the update method for propositional logic, and also a method using Venn-diagrams for syllogistic reasoning. In this chapter we will introduce a uniform method to decide validity for the logics of the first part of this book. This method has been introduced for propositional logic and predicate logic by the Dutch philosopher and logician Evert Willem Beth (1908-1964) in the fifties of the previous century.

The basic idea behind this method comes down to the following principle which we have stressed at earlier occasions. ref

> An inference is valid if and only if there exists *no* counter-examples, i.e., there is no situation in which the premises hold and the conclusion is false.

The method consists of a rule-based construction of a counter-example for a given inference. Each step of the construction is given account of in a tree-like structure which is called a *tableau*. During this construction it may be that, due to conflicting information, the system detects that no counter-examples can be constructed. We speak of a *closed tableau* in such a case, and it implies that no counter-examples exist. We may then safely conclude that the inference which we are analyzing must be valid.



Evert Willem Beth (left). TABLEAUX is a large biannual conference where computer scientists and logicians meet to present and discuss the latest developments on tableau methods and their application in automated reasoning systems.

The tableau method is a very powerful method. It is complete for propositional and predicate logical reasoning. This means that in case of a valid inference the validity can always be proved by means of a *closed* tableau, that is, the exclusion of counter-examples. Moreover, the tableau method can be implemented quite easily within computer programs, and is therefore used extensively in the development of automated reasoning systems.

In the case of propositional logic the tableau method we will discuss here can generate *all* counter-models for invalid inferences. In this respect, the situation in predicate logic is quite different. If an inference is invalid a counter-model must exist, but it may be that it can not be constructed by means of the rules of the tableau system. In this chapter we will introduce two tableau systems for predicate logic of which one is better (but a bit more difficult) than the other in finding counter-models for invalid inferences, but still this more advanced system is not able to specify infinite counter-models, which means that invalid inferences with *only* infinite counter-models — we will see one example in the section on predicate logic — their invalidity can not be demonstrated by this system. In fact, a perfect tableau system does not exist for predicate logic. Since the thirties of the previous century, due to the work of Alonzo Church and Alan Turing, we know that there exists no decision method in general which detects invalidity for all invalid predicate logical inferences.

8.1 Tableaus for propositional logic

But let us first start with the propositional logical case. For checking the validity of a propositional logical inference we can use the method of truth-tables (Chapter 2). If we have an inference $\varphi_1, ..., \varphi_n/\psi$ then we need to set up truth-tables for all the formulas $\varphi_1, ..., \varphi_n, \psi$ and then see whether there is one row, at which the formulas $\varphi_1, ..., \varphi_n$ are

all true (1) and ψ is false (0). If this is the case we have detected a counter-model, and then the inference must be invalid. If such a row can not be found then the inference must be valid since it does not have counter-models in this case.

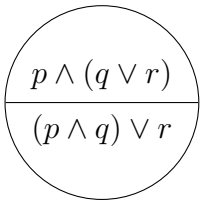
The tables are built up step by step, assigning truth-values to the proposition letters, who represent some atomic bit of propositional information, and then assigning truth-values to all the formulas following the grammatical structures of the formulas. It is therefore called a *bottom up* method.

The tableau method works exactly in the opposite direction: *top-down*. It starts with the original inference and then tries to break it down into smaller pieces. If it arrives at the smallest parts, the proposition letters, and has not run into contradictions then this atomic information can be used to specify a counter-model and invalidity for the given inference has then been proved. If it does not succeed to do so then the tableau is a proof that no counter-model exists, and in this case, the inference must be valid.

Let us get more specific and take a simple valid inference:

$$p \wedge (q \vee r) \models (p \wedge q) \vee r \tag{8.1}$$

We start with a simplistic representation of a candidate counter-example. It depicts a world with two hemispheres of which the true information is contained in the upper half, and the false information in the lower part.



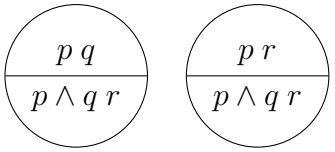
(8.2)

The truth-conditions of the propositions, as defined by the connectives they contain, determine whether this potential counter-example can be realized. As the only true formula is a conjunction, we know that the two conjuncts must be true. The only false proposition is a disjunction, and therefore both these disjuncts must also be false. This leads to the following further specification of our potential counter-example:

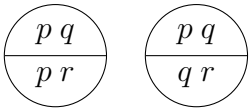
(8.3)

We know now that our candidate counter-example must at least support p and falsify r. The exclusion of six other valuations has already taken place by this simple derivation. Still it is not sure whether the picture in (8.3) captures a real counter-example since $q \vee r$ must be true and $p \wedge q$ must be false. The first formula is a disjunction and because it is true, the formula itself does not give us accurate information about the truth-values of the

the arguments q and r. The only thing we know is that at least one of them must be true. This makes our search more complicated. The following two candidates are then both potential counter-examples.



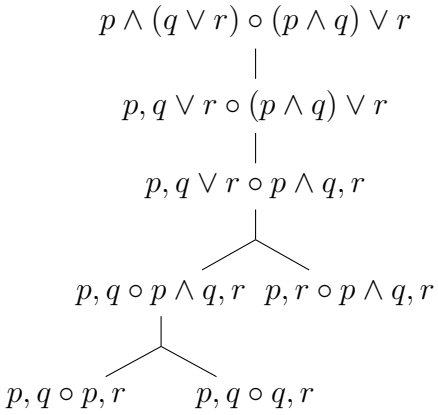
The world on the right hand can not be a counter-example because it requires r to be both true and false. This can never be the case in one single world, and therefore this possibility has to be canceled as a counter-model. The candidate on the left contains a false conjunction, $p \wedge q$. Again, this gives us no precise information, since the falsity of a conjunction only claims the falsity of at least one of the conjuncts. As a consequence, this world must be separated into the following two possibilities.



(8.4)

The first of these two possibilities can not represent a real world because p is both true and false there. The second can not be realized either since q is both true and false in this case. Real counter-examples for this inference do not exist! The inevitable conclusion is that $(p \wedge q) \vee r$ must be true whenever $p \wedge (q \vee r)$ is true.

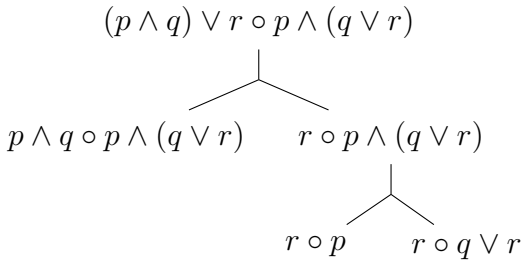
For sake of notation, we will not continue to use the encircled representations as in this first example. We will use a little circle $\circ$ instead and write the *true* formulas on its *left* side, and the *false* formulas on the *right* side of this circle. Doing so, we can summarize our search for a counter-example for the inference $p \wedge (q \vee r)/(p \wedge q) \vee r$ as a tree in the following way.



(8.5)

Each node in the tree is called a *sequent*. A tree of sequents is called a *tableau*. A branch of such a tableau is *closed* if its end node contains a sequent with a formula which appears

both on the left (true) *and* on the right (false) part of the sequent. It means that this branch does not give a counter-example for the sequent as given at the top of the tableau. If all branches are closed then the tableau is also *closed*, and it says, just as in the earlier example, that the top-sequent represents in fact a valid inference. A branch of a tableau is called *open* if its final node is not closed and contains no logical symbols. In this case we have found a counter-example since there are only propositional letters left. A valuation which assigns the value 1 to all the proposition letters on the left part of such a sequent in this end node and 0 to those on the right side will be a counter-model for the inference with which you started the tableau. To illustrate this we can take the earlier example and interchange premise and conclusion. The inference $(p \wedge q) \vee r / p \wedge (q \vee r)$ is an invalid inference, and by using the tableau method we should be able to find a counter-model.



(8.6)

In the first step we have removed the disjunction on the left which led to two possibilities. Then in the second resulting sequent we have removed the conjunction on the right part of the sequent, which led to two new possibilities. The final node with the sequent $r \circ p$ represents a real counter-example. This branch is open. A valuation V with $V(p) = 0$ and $V(r) = 1$ is a counter-model indeed: $V((p \wedge q) \vee r) = 1$ and $V(p \wedge (q \vee r)) = 0$. In fact, this open branch represents two counter-examples since the truth-value of q does not matter in this case. The situations $\overline{pq}r$ and $\overline{p}qr$ are both counter-examples.

The reader may check for himself that the other branches do not give other counter-models. They all close eventually. This means that there are only two counter-models. The tableau as given in (8.6) suffices as a proof of invalidity here. As soon as an open branch has been constructed it is not needed to inspect the other branches.

8.1.1 Reduction rules

A proper tableau needs to be set up according precise *reduction rules*. A reduction rule is specified by the logical symbol that is to be removed, and the truth-value of the formula as indicated by the sequent (left or right of the truth-falsity separation symbol $\circ$). Such a rule is defined by the truth-conditions for the logical symbols. The following schema depicts the rules for conjunction and disjunction, which we already have used in the previous

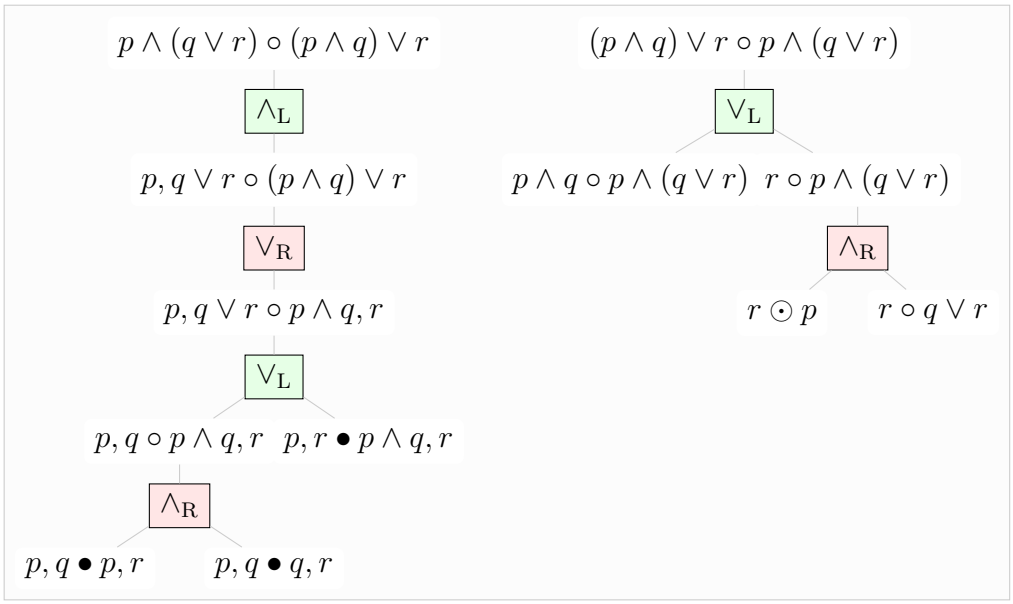
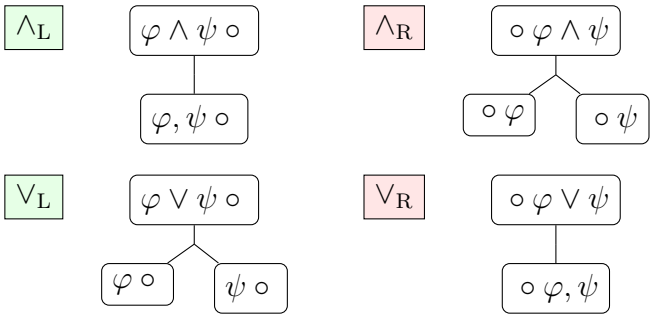


Figure 8.1: Complete tableaus for the earlier examples.

examples.



(8.7)

The rules $\wedge_{\mathrm{L}}$ and $\vee_{\mathrm{L}}$ tell us what to do with a conjunction and disjunction, respectively, when it appears on the left side of a sequent. We use a green background here to make it explicit that when we apply such a rule, we are working on a formula which is claimed to be true. The R-rules are rules which deal with false conjunctions and disjunctions. The background color red is used to stress that we are reducing a formula which is claimed to be false.

In figure 8.1 the earlier examples are given once more, but extended with specifications of the rules we use in each step. As to distinguish open and closed branches we replace the truth-falsity separation symbol $\circ$ by $\odot$ and $\bullet$ respectively. We will continue to use this way of indication in the sequel of this chapter.

For the other connectives rules can be given quite straightforwardly by using the truth-conditions which have been defined in the introductory chapter on propositional logic.

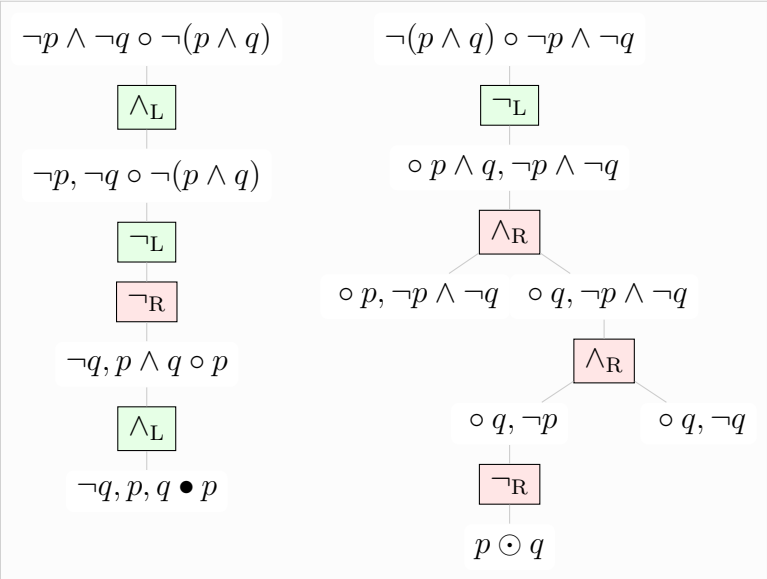
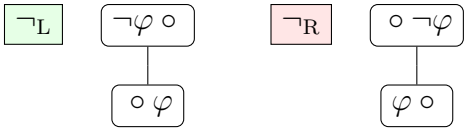


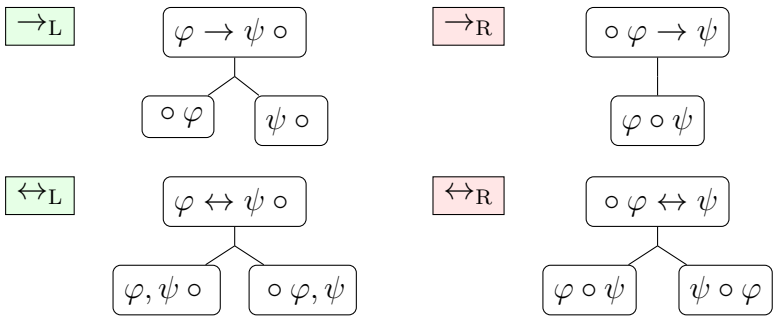
Figure 8.2: Two tableaus with negations. The left tableau shows that $\neg p \wedge \neg q \models \neg(p \wedge q)$. The right tableau shows that the converse of this inference is not valid $\neg(p \wedge q) \not\models \neg p \wedge \neg q$. The counter-model which has been found in the open branch is the valuation which assigns 1 to p and 0 to q. This suffices to show the invalidity of the inference. If we would have worked out the left branch as well we would have found the other counter-example $\overline{p}q$.

The negation rules are the most simple ones. A negation switches truth-values, so the proper way to remove a negation is to transfer its argument from one side of the sequent to the other.



(8.8)

In Figure 8.2 two simple tableaus are given with occurrences of negations. The rules for implication and equivalence are the following:



(8.9)

The rules for equivalence are quite easy to understand. An equivalence is true if the truth-values of the two arguments are the same. In terms of reductions this means that

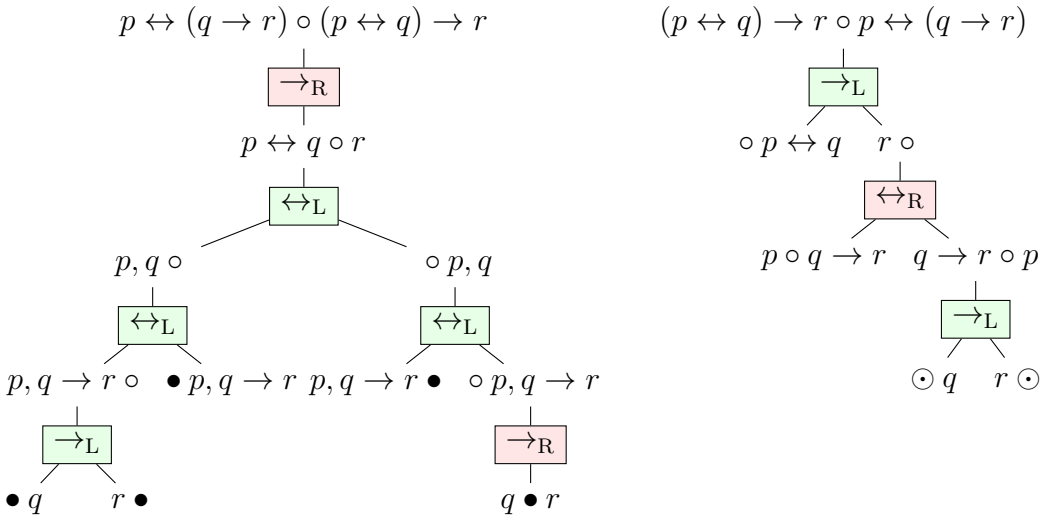
if the equivalence appear on the left hand side of the sequent the two arguments remain on the left hand side (both true) or they switch both to the right hand side (both false). If an equivalence is false the two truth-values of the arguments differ, which gives the two possibilities as captured by $\leftrightarrow_{\mathrm{R}}$-rule as shown in the schema (8.9).

The R-rule for implication captures the only possibility for an implication to be false. The antecedent should be true (moves to the left) and the consequent should be false (stays on the right). The L-rule captures the other possibilities: φ is false (moves to the right) or ψ is true (stays on the left).

Exercise 8.1 Define appropriate reduction rules for the exclusive disjunction $\sqcup$. Remember that $\varphi \sqcup \psi$ is true if and only if exactly one of the arguments φ or ψ is true.

Exercise 8.2 Show that $\neg(\varphi \sqcup \psi)$ is logically equivalent with $\neg\varphi \sqcup \psi$ using the rules that you have defined for $\sqcup$ in the previous exercise. You will need two tableaus here, one for proving that $\neg(\varphi \sqcup \psi) \models \neg\varphi \sqcup \psi$ and one for proving that $\neg\varphi \sqcup \psi \models \neg(\varphi \sqcup \psi)$.

Below in (8.10) two tableaus are given of which the first shows that $p \leftrightarrow (q \to r) \models (p \leftrightarrow q) \to r$. The second demonstrates that the converse is invalid.



(8.10)

For sake of shorter notation we have left out the repetition of formulas, and only kept track of new formulas. This makes it bit harder to read the tableau, but it may be worth the effort to get used to this shorter notation since tableaus, especially in the case of predicate logic as we will see in the next section, tend to get very large.

In order to conclude closure of a branch we need to scan it for contradiction in backward direction. This also is the case for defining a counter-model for an open branch. For example, the counter-examples as given in the right-most branch in the second tableau of (8.10) are those who falsify p and verify r. About the proposition letter q no information is given in this open branch.

Exercise 8.3 Use tableaus to test the validity of the following inferences.

(1) $p \vee (q \wedge r)/(p \vee q) \wedge (p \vee r)$

(2) $p \to q, q \to r/\neg r \to \neg p$

Exercise 8.4 Use the tableau method to find out whether the following sets of formulas are consistent (satisfiable), i.e., check whether there is a valuation which makes all the formulas in the given set true.

(1) $\{p \leftrightarrow (q \vee r), \neg q \to \neg r, \neg(q \wedge p), \neg p\}$

(2) $\{p \vee q, \neg(p \to q), (p \wedge q) \leftrightarrow p\}$

Exercise 8.5 Check, using the tableau method, whether the following formulas are tautologies or not.

(1) $(p \to q) \vee (q \to p)$

(2) $\neg(p \leftrightarrow q) \leftrightarrow (\neg p \leftrightarrow \neg q)$

8.2 Tableaus for predicate logic

A tableau system consists of the rules for the connectives as given in the previous section and four rules for the quantifiers, two rules for each of the two quantifiers $\forall$ and $\exists$.[1] These rules are a bit more complicated because the quantifiers range over the individual objects in the domain of the models. Beforehand, however, we do not know how many of those individuals are needed to provide real counter-models. The domain has to be constructed step by step. This makes it harder to process universal information adequately because it needs to be applied to *all* the objects, and it may be that it will not be clear at that stage what the required set of objects is to provide a counter-example. In simple cases this can be avoided by dealing with the existential information first. Let us have a look at such an easy going example:

$$\forall x\,(Px \vee Qx)/\forall x\,Px \vee \forall x\,Qx \tag{8.11}$$

It may be clear that this an invalid inference. Every integer is even or odd ($P \vee Q$) but it is surely not the case that all integers are even (P) or that all integers are odd (Q). Let us see what happens if we want to demonstrate this with a tableau. At first we can apply the $\vee_{\mathrm{R}}$-rule as we have defined in the previous section:

$$\begin{array}{c} \forall x\,(Px \vee Qx) \circ \forall x\,Px \vee \forall x\,Qx \\ \big| \\ \boxed{\vee_{\mathrm{R}}} \\ \big| \\ \forall x\,(Px \vee Qx) \circ \forall x\,Px, \forall x\,Qx \end{array} \tag{8.12}$$

[1] For sake of keeping things simple, we will not deal with the equality sign $=$ and function symbols here. Moreover, we assume that all formulas contain no free variables.

For the potential counter-model this means the following. All individuals are $P \vee Q$-s but not all of them are P-s and not all of them are Q-s, since $\forall x\, Px$ and $\forall x\, Qx$ must be falsified. They occur on the right side of the last sequent. A universally quantified formula $\forall x\, \varphi$ on the right hand side of a sequent conveys an *existential* claim, we need at least one non-φ-er within the candidate counter-model. As we said earlier, it is better to deal with this existential information first. Removal of the formula $\forall x\, Px$ can be done by replacing it by Pd_1 where d_1 is some additional name for the object which does not have the property P. We do not know who or what this non-P-object is, and therefore we need a neutral name to denote it. So our next step is:

$$\begin{array}{c} \forall x\,(Px \vee Qx) \circ \forall x\, Px, \forall x\, Qx \\ \hline \forall_{\mathrm{R}} \\ \forall x\,(Px \vee Qx) \circ Pd_1, \forall x\, Qx \end{array} \tag{8.13}$$

Elimination of the last universal quantifier on the right hand side requires a non-Q-object. This object may be different from d_1 and therefore we choose a new neutral name d_2.[2]

$$\begin{array}{c} \forall x\,(Px \vee Qx) \circ Pd_1, \forall x\, Qx \\ \hline \forall_{\mathrm{R}} \\ \forall x\,(Px \vee Qx) \circ Pd_1, Qd_2 \end{array} \tag{8.14}$$

At this stage we have to eliminate the universal quantifier on the left hand side of the sequent. We need to apply the property $Px \vee Qx$ to all the objects in the domain. This far we only have objects called d_1 and d_2 and therefore we *only* apply it to those objects, which brings two new formulas on the stage $Pd_1 \vee Qd_1$ and $Pd_2 \vee Qd_2$. In this case we are sure that no other objects may be needed because all the existential information has been dealt with in the two steps before.

$$\begin{array}{c} \forall x\,(Px \vee Qx) \circ Pd_1, Qd_2 \\ \hline \forall_{\mathrm{L}} \\ Pd_1 \vee Qd_1, Pd_2 \vee Qd_2 \circ Pd_1, Qd_2 \end{array} \tag{8.15}$$

[2]Note that we do not exclude the possibility that d_1 and d_2 are equal here. In predicate logic it is possible that one object carries two names.

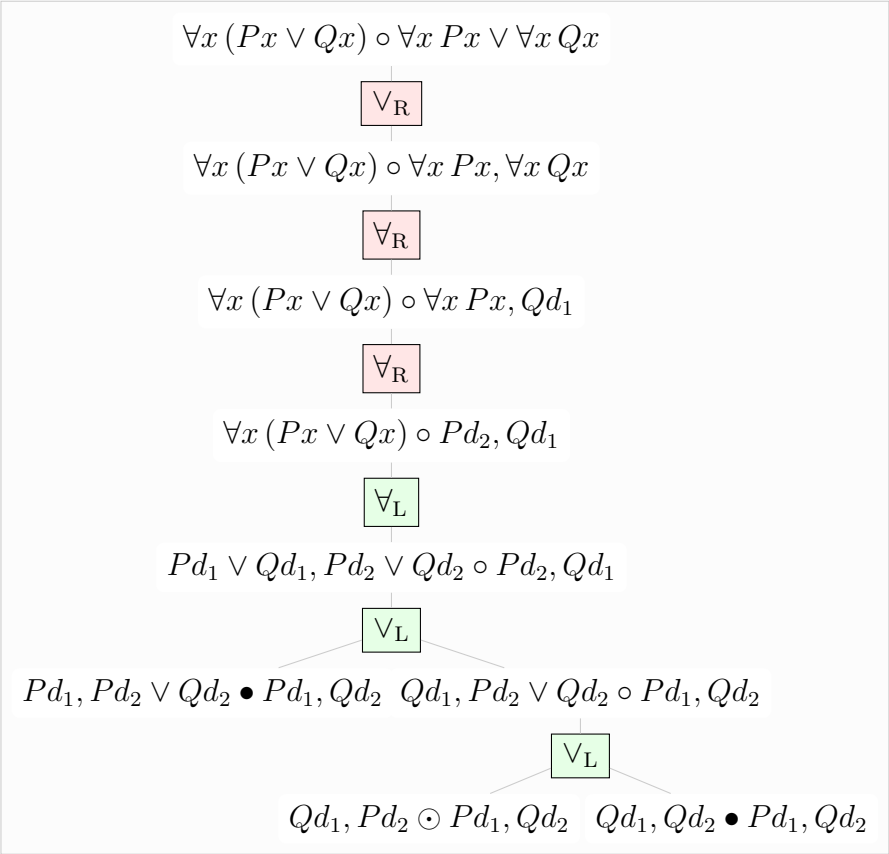
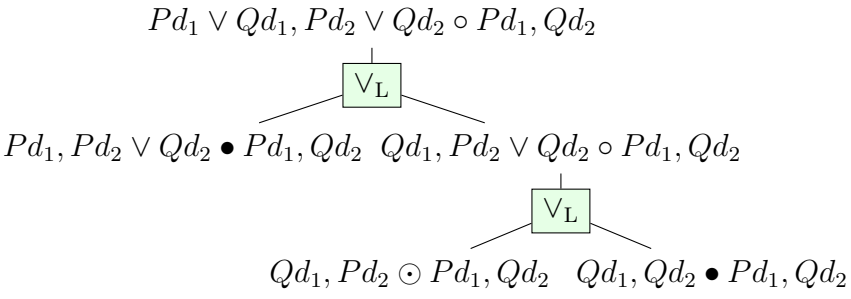


Figure 8.3: The full tableau demonstrating that $\forall x\,(Px \vee Qx) \not\models \forall x\,Px \vee \forall x\,Qx$. The counter-example contains a P who is not Q and a Q who is not P.

The last two steps deal with the two disjunctions.



(8.16)

Finally, we have found a counter-model. The open branch tells us that we need a model with two objects. The first one needs to be a Q-object which does not have the property P, and the second has to be a P-object which does not have the property Q. This is indeed a counter-model for the original inference as given in (8.11). In Figure 8.3 the full tableau is given.

Exercise 8.6 Show with a tableau that $\exists x\,(Px \wedge Qx) \models \exists x\,Px \wedge \exists x\,Qx$.

Exercise 8.7 Show with a tableau that $\exists x\,Px \wedge \exists x\,Qx \not\models \exists x\,(Px \wedge Qx)$.

Exercise 8.8 Show with a tableau that $\forall x\,(Px \vee Qx) \models \forall x\,Px \vee \exists x\,Qx$.

In the last example we dealt with existential information before we used the universal information. This is not always possible. Here is a short but more complicated case.

$$\exists x\,(Px \to \forall y\,Py) \tag{8.17}$$

The formula says that there exists an object such that if this object has the property P then every object has the property P. We do not know which object is meant here so let us give it a neutral name. The formula then reduces to $Pd_1 \to \forall y\,Py$. Such an object can then always be chosen. If all objects have the property P then it does not matter which object you choose, since the consequent is true in this case. If, on the other hand, not all objects have the property P then you can pick one of the non-P-objects for d_1. The antecedent is then false and therefore the implication $Pd_1 \to \forall y\,Py$ holds. In other words, $\exists x\,(Px \to \forall y\,Py)$ is *valid*.

In order to prove that (8.17) is valid by means of a tableau we have to show that it never can be false. Putting it on the right side of the top-sequent, we then should be able to construct a closed tableau. Here is a first try in three steps.

$$\begin{array}{c} \circ\ \exists x\,(Px \to \forall y\,Py) \\ \big|\ \exists_{\mathrm{R}} \\ \circ\ Pd_1 \to \forall y\,Py \\ \big|\ \to_{\mathrm{R}} \\ Pd_1 \circ \forall y\,Py \\ \big|\ \forall_{\mathrm{R}} \\ Pd_1 \circ Pd_2 \end{array} \tag{8.18}$$

Foremost, we need to explain the first step. An existential quantified formula on the right yields a universal claim. If $\exists x\,\varphi$ is false it means that there exists *no* φ-er: φ is false for all individuals in the domain. Since there are no objects introduced so far the reader may think that this leads to an empty sequent. But in predicate logic we have a minimal convention that every model has at least one object.This means that if we want to fulfill a universal claim, that is, a true formula of the form $\forall x\,\varphi$ or a false formula of the form $\exists x\,\varphi$, and there are no objects introduced so far then we introduce one. This is what has been done in the first step in (8.18).

The second and the third step are as before. Now, it may seem as if we have an open branch here since there is no contradictory information and there are no logical symbols left. But we made a logistic mistake here. We removed the false formula $\forall y\, Py$ here by introducing a new non-P-object called d_2. The universal claim by the false formula $\exists x\,(Px \to \forall y\, Py)$ however has been applied to d_1 only, whereas $Px \to \forall y\, Py$ has to be false for *all* objects, and therefore, also for d_2! In tableau-systems for predicate logic this means that whenever a new name is to be introduced the formulas which have universal strength which have been removed at an earlier stage in the tableau will become active again, and then need to be dealt with at a later stage. So the last step of (8.18) need to be extended in the following way:

$$\begin{array}{c} Pd_1 \circ \forall y Py \\ \hline \forall_{\mathrm{R}} \\ \hline Pd_1 \circ Pd_2, \exists x\,(Px \to \forall y\, Py) \end{array} \tag{8.19}$$

The formula $\exists x\,(Px \to \forall y\, Py)$ is supposed to be falsified in the end, and becomes active again when the new object called d_2 is introduced. The next step then is to deny the property $Px \to \forall y\, Py$ for all objects. Since it has been already denied for d_1 in the first step in (8.18), the only new information is that $Pd_2 \to \forall y\, Py$ must be false.

$$\begin{array}{c} Pd_1 \circ Pd_2, \exists x\,(Px \to \forall y\, Py) \\ \hline \exists_{\mathrm{R}} \\ \hline Pd_1 \circ Pd_2, Pd_2 \to \forall y\, Py \end{array} \tag{8.20}$$

One may think, at first sight, that this leads to an infinite procedure. In this case, things work out nicely, since the tableau closes in the next step. The implication will be removed, and then we run into a conflict: Pd_2 must be true and false at the same time.

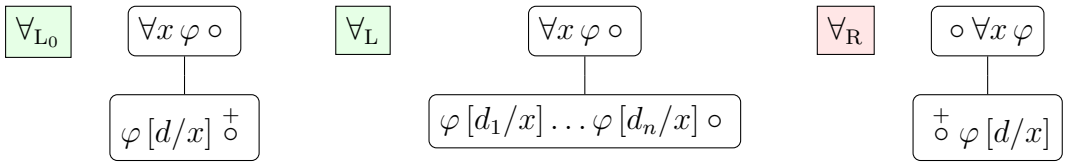
$$\begin{array}{c} Pd_1 \circ Pd_2, Pd_2 \to \forall y\, Py \\ \hline \to_{\mathrm{R}} \\ \hline Pd_1, Pd_2 \bullet Pd_2, \forall y\, Py \end{array} \tag{8.21}$$

This means that there are no models which falsify $\exists x\,(Px \to \forall y\, Py)$. This formula must be valid.

8.2.1 Rules for quantifiers

In the two examples above we have indicated how we should deal with quantified predicate logical formulas in a tableau. Here we want to give a formal status to the reduction

rules for the quantifiers. Let us start with the universal quantifier.



(8.22)

There are two left rules for the universal quantifier when it appears on the left part of a sequent.

$\forall_{L_0}$: The first rule (0) is meant to deal with the exceptional case when no names are present in the sequent, that is, there are no names to apply the property φ to. In this case we introduce a new name d and replace all free occurrences of x in φ by d. We write this as $\varphi\,[d/x]$. In addition, the truth-falsity separation symbol $\circ$ is designated with a $+$ on top to indicate that a *new* name has been added within the branch of the tableau.

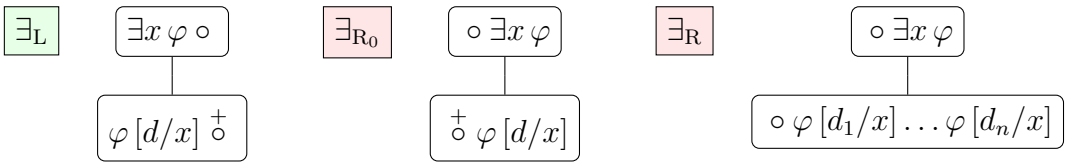
$\forall_{L}$: If names are present in the input sequent then $\forall x\,\varphi$ can be removed from the left part of the sequent by applying φ to the names $d_1, ..., d_n$, all occurring in the sequent and which φ has *not* been applied to *yet*.

$\forall_{R}$: A false formula $\forall x\,\varphi$ is removed by applying φ to a new name d. This denotes the object we need as an example of a non-φ-er in the counter-model which we are constructing. In order to show that this name is new we use the additional $+$-indication.

In the end we need to distinguish $\circ$ from $\overset{+}{\circ}$-sequents in which new name are introduced.

$\overset{+}{\circ}$: If a new name is introduced then all formulas of the form $\forall x\,\varphi$ appearing on the left part and those of the form $\exists x\,\varphi$ on the right part of preceding sequents in this branch re-appear in the output sequent.

The rules for the existential quantifiers are defined analogously to the rules for the universal quantifier:



(8.23)

The following example, which shows that $\exists y \forall x\, Rxy \models \forall x \exists y\, Rxy$, make use of all the general rules.

$$\begin{array}{c} \exists y \forall x\, Rxy \circ \forall x\, \exists y\, Rxy \\ \Big| \\ [\exists_{\mathrm{L}}] \\ \Big| \\ \forall x\, Rxd_1 \overset{+}{\circ} \forall x\, \exists y\, Rxy \\ \Big| \\ [\forall_{\mathrm{R}}] \\ \Big| \\ \forall x\, Rxd_1 \overset{+}{\circ} \exists y\, Rd_2y \\ \Big| \\ [\forall_{\mathrm{L}}] \\ \Big| \\ Rd_1d_1, Rd_2d_1 \circ \exists y\, Rd_2y \\ \Big| \\ [\exists_{\mathrm{R}}] \\ \Big| \\ Rd_1d_1, Rd_2d_1 \bullet Rd_2d_1, Rd_2d_2 \end{array} \tag{8.24}$$

In this example the quantifiers were in optimal position. We could fulfill the existential claims ($\exists_{\mathrm{L}}$ and $\forall_{\mathrm{R}}$) before we dealt with the universal requirements ($\forall_{\mathrm{L}}$ and $\exists_{\mathrm{R}}$) for the potential counter-model. As a result of this no reintroduction of universal information was needed.

In (8.18) we already have seen that this reintroduction can not always be avoided. Fortunately, this did not lead to an infinite procedure, because the tableau could be closed. But in other cases we may run into real trouble due to continuing introduction of new names, and consequently, unstoppable re-appearance of universal information. Below such an example is given. Let us first look at the first two steps.

$$\begin{array}{c} \forall x\, \exists y\, Rxy \circ \exists y \forall x\, Rxy \\ \Big| \\ [\forall_{\mathrm{L}_0}] \\ \Big| \\ \exists y\, Rd_1y \overset{+}{\circ} \exists y \forall x\, Rxy \\ \Big| \\ [\exists_{\mathrm{L}}] \\ \Big| \\ \forall x \exists y\, Rxy, Rd_1d_2 \overset{+}{\circ} \exists y \forall x\, Rxy \end{array} \tag{8.25}$$

The two formulas in the top-sequent have universal status. The left formula is true and says that every object is R-related to some object. In a domain of persons, taking the relation Rxy to represent the relation 'x loves y', $\forall x \exists y\, Rxy$ means "Everybody loves

somebody". The formula $\exists y \forall x\, Rxy$ on the right hand should be falsified, and therefore the claim is that is not the case that there exists an object such that all objects are R-related to it. In the context mentioned here above, this means that there is no person who is loved by everybody. So, there is no other option than to apply one of the exceptional universal rules $\forall_{\mathrm{L}_0}$ or $\exists_{\mathrm{R}_0}$. We have chosen to take the former.

In the second step we took the new existential formula on the left since we prefer to deal with existential information first. Here we introduced a new name, and therefore, the universal formula which has been removed in the first step pops up again. Repetition of the same procedure would introduce a third object and a second re-appearance of $\forall x \exists y\, Rxy$. If, instead, we would choose to remove the formula $\exists y \forall x\, Rxy$ on the right we would then get the following two successive steps:

$$\begin{array}{c} \forall x\, \exists y\, Rxy, Rd_1d_2 \circ \exists y \forall x\, Rxy \\ \downarrow \exists_{\mathrm{R}} \\ \forall x\, \exists y\, Rxy, Rd_1d_2 \circ \forall x\, Rxd_1, \forall x\, Rxd_2 \\ \downarrow \forall_{\mathrm{R}} \\ \forall x\, \exists y\, Rxy, Rd_1d_2 \overset{+}{\circ} Rd_3d_1, \forall x\, Rxd_1, \exists y \forall x\, Rxy \end{array} \tag{8.26}$$

In the last step a third object is introduced, and then $\exists x \forall y\, Rxy$ re-appears on the right part of the sequent. The sequent in the last node contains the same formulas as in the top node with two additional atomic formulas who do not contradict each other. Moreover, we know that this tableau will never close since the top sequent represents an invalid inference. This branch will *never* end with the desired final sequent free of logical symbols.

Without applying the rules it is not hard to find a simple counter-example. Take the situation of two persons who love themselves but not each other. In such a case, $\forall x \exists y\, Rxy$ is true and $\exists y \forall x\, Rxy$ is false, since there is no person who is loved by everybody. Apparently, our tableau system is not able to find such a simple counter-model. In fact the rules guide us towards an *infinite* counter-example which can never be constructed since in each step at most one additional object is introduced.

Despite this inability of the system, the rules make up a complete validity testing method. If an inference $\varphi_1, ...\varphi_n/\psi$ is valid, $\varphi_1, ..., \varphi_n \models \psi$, then there exists a closed tableau with $\varphi_1, ..., \varphi_n \circ \psi$ as the top sequent. We will not prove this completeness result here, but instead, get into more detail at a later stage.

Exercise 8.9 Test the validity of the following syllogisms with tableaus:

(1) $\forall x\,(Ax \to Bx), \exists x\,(Ax \wedge Cx)/\exists x\,(Cx \wedge Bx)$

(2) $\forall x\,(Ax \to Bx), \exists x\,(Ax \wedge \neg Cx)/\exists x\,(Cx \wedge \neg Bx)$

(3) $\neg\exists x\,(Ax \wedge Bx), \forall x\,(Bx \to Cx) / \neg\exists x\,(Cx \wedge Ax)$

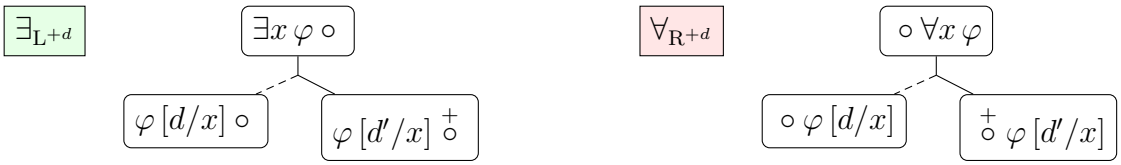
Exercise 8.10 Prove the validity of the following inference with tableaus:

(1) $\forall x\,(Ax \to Bx) \vee \forall y\,(By \to Ay) \models \forall x\,\forall y\,((Ax \wedge By) \to (Bx \vee Ay))$

(2) $\forall x\,\forall y\,((Ax \wedge By) \to (Bx \vee Ay)) \models \forall x\,(Ax \to Bx) \vee \forall y\,(By \to Ay)$

8.2.2 Alternative rules for finding finite counter-models

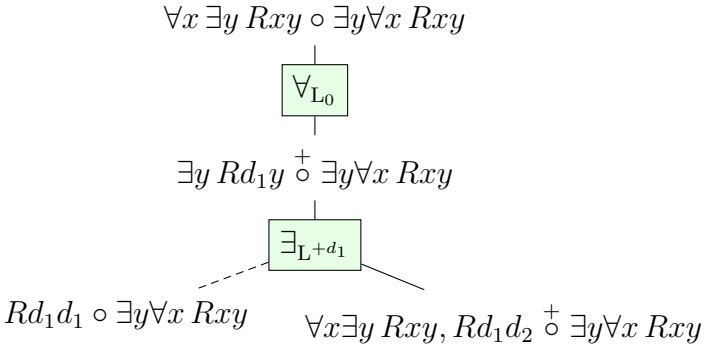
In (8.25) and (8.26) we have seen an example of an invalid inference with quite simple finite counter-models which can not be found by means of the rules for the quantifiers. In order to find such finite counter-models with a tableau system we need to extend the rules for the quantifiers a bit. The problem with the earlier rules was the introduction of new names which caused repetitive reintroduction of formulas. This can be avoided by facilitating the 'old' objects to support existential information. These extended versions of the 'existential' rules $\exists_{\mathrm{L}}$ and $\forall_{\mathrm{R}}$ have the following general format, where the name d is some name which occurs in the input node and d' does not.



(8.27)

The truth-falsity separation sign $\circ$ only has a $+$-sign in the right branch. In the left branch we have used an old object called d which does not provoke reintroduction of universal information. We indicate these special *try-out branches* with a dashed line.

Let us try these extended rules to find a simple finite counter-model for the example we started with in (8.25). Here are the first two steps.



(8.28)

The first step is the same as in (8.25). The second step is the application of the extended version of $\exists_{\mathrm{L}}$. We apply in the left branch the property $Rd_1 y$ to the only known name d_1.

In this branch the true formula $\forall x\,\exists y\,Rxy$ is *not* reintroduced. This try-out branch can then be extended with the following four steps.

$$
\begin{array}{c}
Rd_1d_1 \circ \exists y \forall x\, Rxy \\
\big|\, \exists_{\mathrm{R}} \\
Rd_1d_1 \circ \forall x\, Rxd_1 \\
\big|\, \forall_{\mathrm{R}} \\
\forall x \exists y\, Rxy, Rd_1d_1 \overset{+}{\circ} Rd_2d_1, \exists y \forall x\, Rxy \\
\big|\, \forall_{\mathrm{L}} \\
\exists y\, Rd_2y, Rd_1d_1 \circ Rd_2Rd_1, \exists y \forall x\, Rxy \\
\big|\, \exists_{\mathrm{L}^{+d_2}} \\
Rd_2d_2, Rd_1d_1 \circ Rd_2d_1, \exists y \forall x\, Rxy \qquad \forall x \exists y\, Rxy, Rd_2d_3, Rd_1d_1 \overset{+}{\circ} Rd_2d_1, \exists y \forall x\, Rxy
\end{array}
\tag{8.29}
$$

In the second step we did not apply $\forall_{\mathrm{R}^{+d_1}}$ but the old version $\forall_{\mathrm{R}}$ instead. A try-out branch would close immediately because of the true formula Rd_1d_1. In the last step we have chosen for $\exists_{\mathrm{L}^{d_2}}$. The d_1-version would have given a closed branch because of the false formula Rd_2d_1. Extension of this new try-out branch results into our desired counter-model in two more steps.

$$
\begin{array}{c}
Rd_2d_2, Rd_1d_1 \circ Rd_2d_1, \exists y \forall x\, Rxy \\
\big|\, \exists_{\mathrm{R}} \\
Rd_2d_2, Rd_1d_1 \circ Rd_2d_1, \forall x\, Rxd_2 \\
\big|\, \forall_{\mathrm{R}^{+d_1}} \\
Rd_2d_2, Rd_1d_1 \odot Rd_2d_1, Rd_1d_2 \qquad \forall x\, \exists y\, Rxy \circ Rd_3d_2, \exists y \forall x\, Rxy
\end{array}
\tag{8.30}
$$

In the first step the false formula $\exists y \forall x\, Rxy$ results into $\forall x\, Rxd_2$ only because $\forall x\, Rxy$ has been applied to d_1 in the third step of this branch (8.29). In the second step we used a d_1-try-out branch. The d_2-variant would have given closure because of the true formula Rd_2d_2. This third try-out branch has finally determined a counter-example. The objects called d_1 and d_2 are R-related to themselves but are mutually not R-related.

It is not hard to see that the d_1-try-out branch in the second step of this tableau (8.28) can not give any other counter-examples with only two objects. If we would have chosen for

the regular branch after this second step we could have constructed the other two object counter-model, that consists of two objects who are mutually related but are not related to themselves. We leave this as an exercise to the reader.

Exercise 8.11 Try to find the other counter-model as mentioned here above using the try-out branches on other places.

Exercise 8.12 Show the invalidity of the following inference with a tableau dressed up with try-out branches. Try to keep the final counter-model as small and simple as possible.

(1) $\forall x \exists y\, Rxy / \forall x \exists y\, Ryx$

(2) $\exists x \forall y\, Rxy / \exists x \forall y\, Ryx$

8.2.3 Invalid inferences without finite counter-examples

With these new extended 'existential' rules we can always find finite counter-examples, but this does not mean that every invalid inference can be recognized as such by the extended tableau system. In predicate logic we can make up invalid inferences with only infinite counter-models. Here is an example with two premises:

$$\forall x \exists y\, Rxy, \forall x \forall y \forall z\, ((Rxy \wedge Ryz) \rightarrow Rxz) \not\models \exists x \exists y\, (Rxy \wedge Ryx) \tag{8.31}$$

Take again the 'love'-interpretation for the relation R then the inference can be rephrased as follows:

Everybody loves somebody
Everybody loves all persons who are loved by his loved ones.

There is at least a pair of persons who love each other. (8.32)

We would expect that the seemingly cautious conclusion would follow from the happy hippie optimism conveyed by the two premises. And in fact it holds as long as we would stick to situations with only a finite number of persons.

Exercise 8.13 Show that for finite models which satisfy the two premises as given (8.31) will always contain a symmetric pair: $\exists x \exists y\, (Rxy \wedge Ryx)$. A finite happy hippie community always contains a happily loving couple!

In situations with an infinite number of objects we can interpret R in such a way that the two premises are true and the conclusion is false. For example, take the integers instead of people with R interpreted as the relation $<$. The inference then can be recaptured as follows:

Every integer is smaller than some integer

If one integer is smaller than a second one, then all the integers which are larger than the second are also larger than the first.

There is at least one pair of integers who are smaller than each other.

Here the premises are clearly true, and the conclusion is false, which proves that the inference as given in (8.31) is indeed invalid. Such infinite counter-models can never be constructed by our tableaus, and since the inference of (8.31) has only infinite counter-examples its invalidity can never be demonstrated by the system, not even with the help of the extended 'existential' rules.

8.2.4 Tableaus versus natural reasoning

Although the tableau systems which we have discussed so far are pretty good computational methods for testing the validity of inferences, there is also a clear disadvantage. Each of the individual steps in a tableau can be understood quite easily, but a tableau as a full line of argumentation is not very transparent, and it seems that it does not reflect the way ordinary humans reason.

One part of this type of objection against the tableau method is superficial because there are many small steps in a tableau that are never made explicitly in a 'normal' argumentation because they are too trivial to be mentioned. Due to the fact that the tableau method is a pure symbolic method all the small steps have to be taken into account, and therefore these steps may look quite artificial.

But there is more to it. In a tableau we reason in a negative way, which tends to be unnatural. Validity is demonstrated by the exclusion of counter-examples, whereas humans tend to prove the validity of an inference directly, from the given facts to what has to be proven. If this does not work, or if uncertainty about the validity of an inference arises, one tries to use his imagination to think of a counter-example to refute the validity. Proof and refutation are most often considered to be two different sorts of mental activity. The tableau method incorporates the two in one computational system by argumentation in an indirect way.

In one of the next chapters we will discuss another proof system which can only be used to demonstrate validity and which makes use of rules which are close to the way humans reason. Therefore, these system are referred to as 'natural deduction'. The tableau systems are considered as 'unnatural deduction'. They are very useful for 'black box' automated reasoning systems, where the users are only interested in a final answer about the validity of an inference (and maybe also a specification of a counter-example) but not *how* it has been computed.

This is quite an exaggerated qualification. It is true that tableau systems may behave in an unnatural way. We have seen an example of the first tableau system for predicate

logic where the system tried to construct a complicated infinite counter-example for an invalid inference for which everybody can make a very simple counter-example. We have also shown how the rules can be 'tamed' to perform in a more 'reasonable' way. The development of more natural and more 'intelligent' tableau systems is an important issue of contemporary research of applied computational logic.

Moreover, systems which may perform strangely in certain circumstances may behave remarkably intelligent in others. Here is an example:

$$\exists x \forall y \left(Rxy \leftrightarrow \neg \exists z \left(Ryz \wedge Rzy \right) \right) \tag{8.33}$$

This is a predicate logical formulation of what is known as the Quine paradox (after the American philosopher and logician Willard Van Orman Quine). This formula turns out to be inconsistent. This is not directly obvious. It really takes some argumentation.

Exercise 8.14 The formula $\exists x \forall y \left(Ryx \leftrightarrow \neg Ryy \right)$ is a predicate logical version of the famous Russell paradox (take R to be the relation 'element of' to get the exact version). Show with a tableau that this formula can never be true (is inconsistent).

The subformula $\neg \exists z \left(Ryz \wedge Rzy \right)$ of (8.33) means that there is no object which is mutually related to y. In a domain of persons and by interpreting Rxy as that 'x knows y' this means that y has no *acquaintances*, that is, persons known by y who also know y. Let us say that such a person without acquaintances is called a 'loner.' The full formula as given in (8.33) says that there exists some person who knows all the loners and nobody else. Let us call this person the 'loner-knower'. According to the following infallible argumentation this loner-knower cannot exist.

- If the loner-knower knows himself, then he has an acquaintance, and therefore he is not a loner himself. But then he knows a person who is not a loner, which contradicts the fact that he only knows loners.
- If he does not know himself, then he is not a loner, otherwise he would know himself. But if he is not a loner then he must have an acquaintance. This acquaintance is a loner neither, since he knows the loner-knower and the loner-knower knows him. So, the loner-knower knows a non-loner, which also contradicts the fact that the loner-knower only knows loners.
- The inevitable conclusion is that the loner-knower does not exist, because for every person it must be the case that he knows himself or not. For the loner-knower both options lead to a contradiction.

In Figure (8.4) on page 8-23 the inconsistency of (8.33) has been demonstrated by means of a closed tableau. This closed tableau, starting with a sequent with the formula on the left hand side, shows that the Quine paradox can never be true. If we rephrase the information in the closing tableau of Figure 8.4 we get in fact quite the same line of

argumentation as have been outlined here above. In Figure 8.5 on page 8-24 the tableau has been translated back to the interpretation in natural language as has been described here.

8.3 Tableaus for epistemic logic

The tableau method is used extensively for classical logics such as propositional and predicate logic. It does not have been applied very much in the field of modal logic, such as the epistemic and dynamic logics that have been introduced in the first part. In modal logic the tradition tends much more towards axiomatic systems and model-checking. Part of the underlying reasons are purely cultural, but there are also important technical reasons. There are many different modal logics using a wide variety of different kind of modal operators. On top of that, these logics also make use of different classes of possible world models. Technically, it is just easier to capture these differences by means of axioms. It is just a matter of replacing some axioms by others in order to skip from one modal logic to the other. Directed search methods, such as the tableau method, are much harder to modify appropriately.

We will avoid technical details here. In order to illustrate a tableau method for a modal system we take the simplest one of which rules look very much like those of the last system we have presented for predicate logical reasoning. The system we will discuss here is a validity test for inferences in epistemic propositional logic with only one agent, that is, propositional logic with a single K operator.

Remember that $K\varphi$ stands for the proposition which says the agent knows that φ is the case, and in terms of possible world semantics, it meant that φ is true in all the agent's epistemic alternatives. This means that we have to keep track of more worlds in one node in a tableau, since a counter-model may exist of multiple worlds. In tableau terminology these are called *multi-sequents* which are represented by boxes which may contain more than one sequent.

Below the rules have been given in a brief schematic way. The vertical lines of dots represent one or more sequents, and $\varphi >$ means that the formula φ appears on the left hand side of at least one of the sequents in the box, and $\varphi \{$ means it appears in all of them. The symbols $< \varphi$ and $\{ \varphi$ are used to describe analogous situations for formulas on the right side.

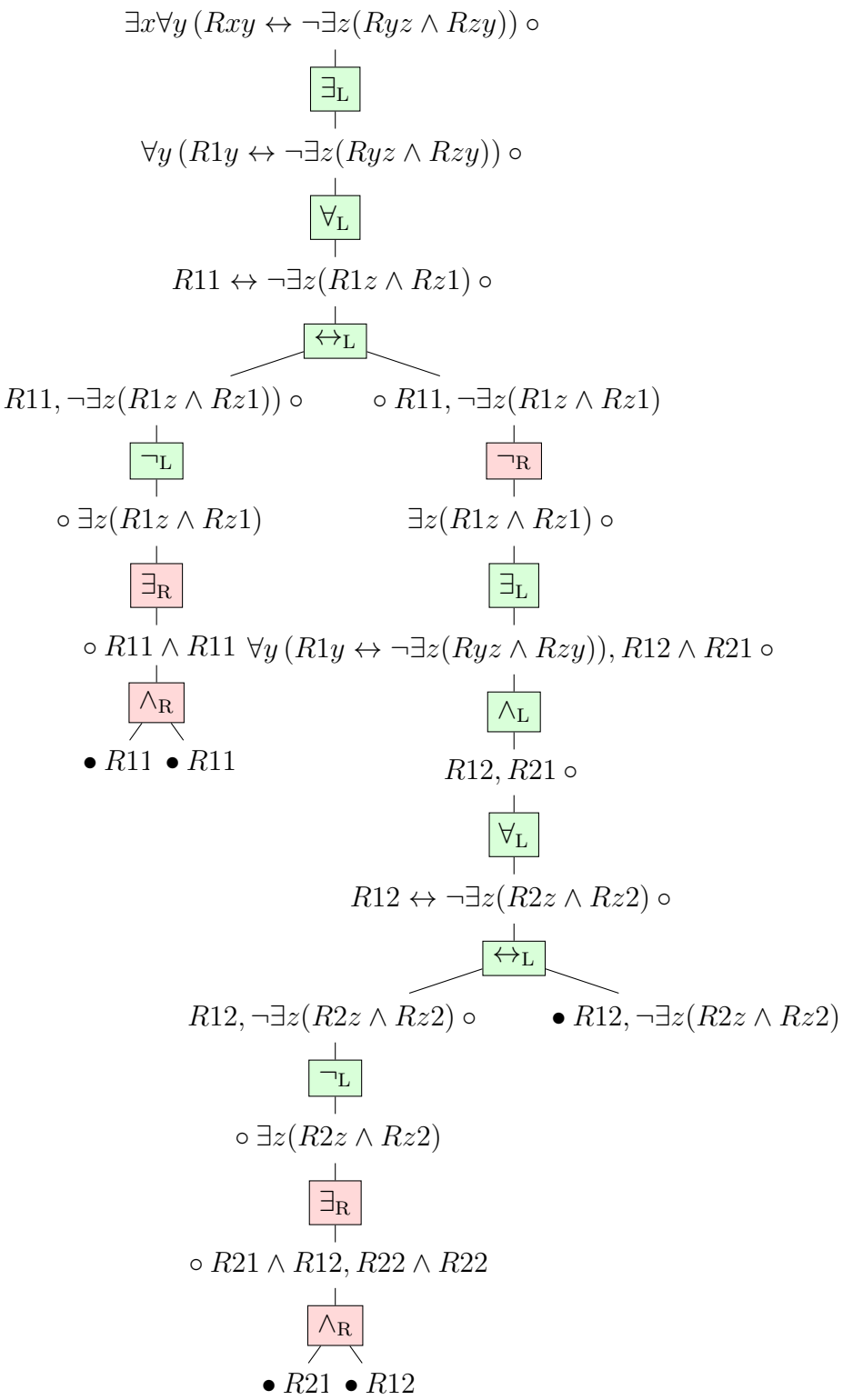


Figure 8.4: A tableau which proves that the Quine-paradox is not satisfiable.

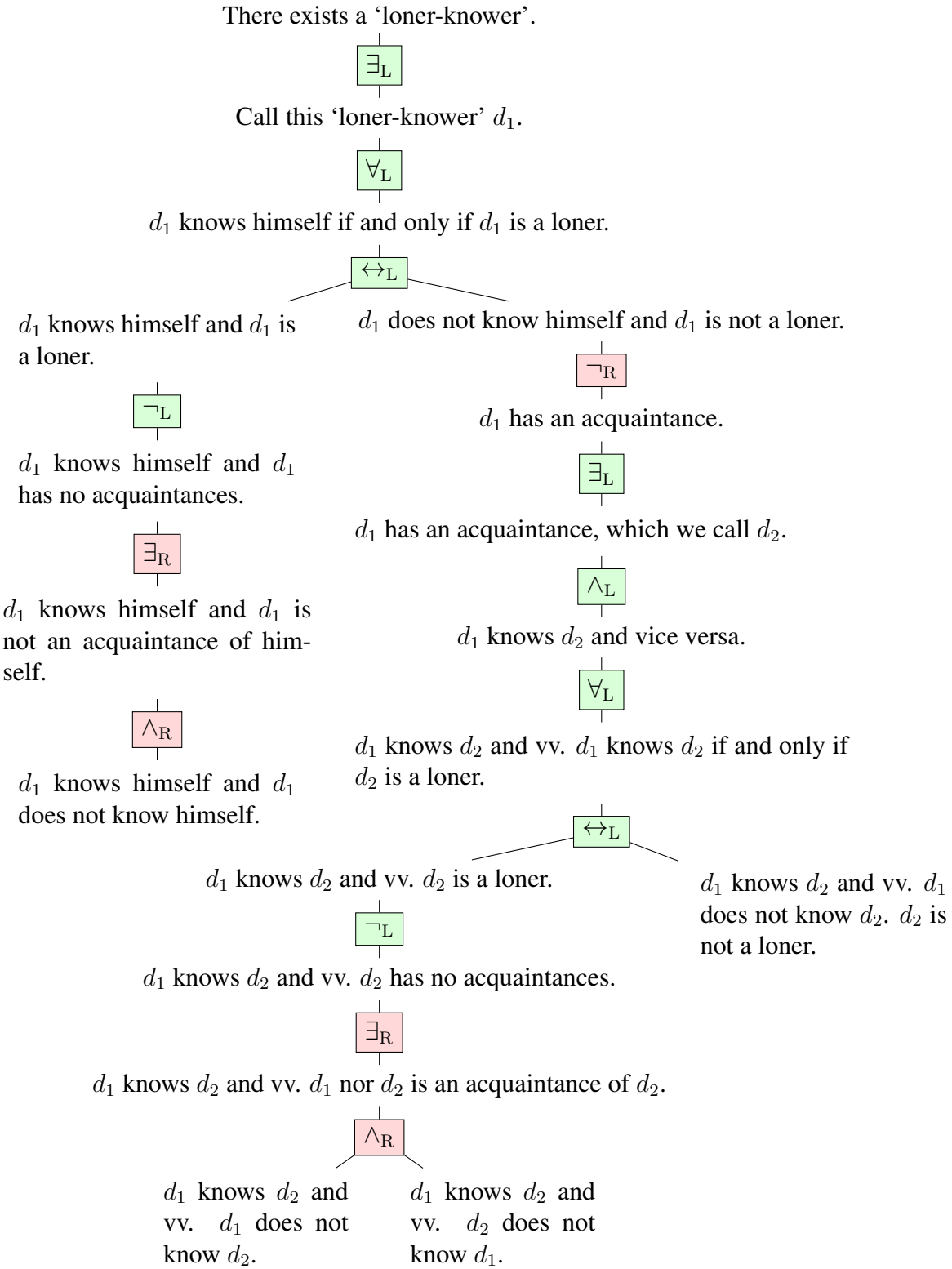
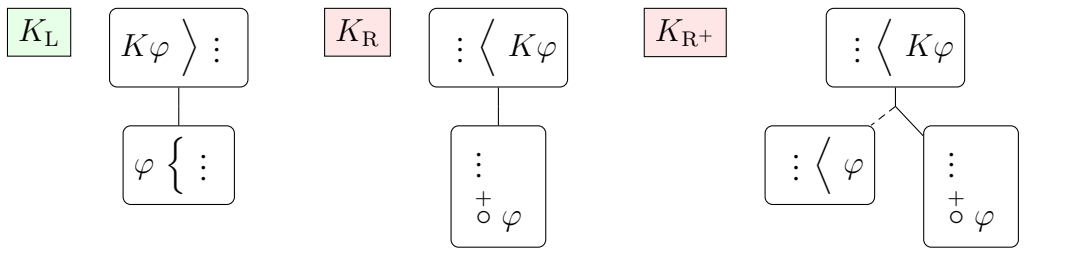


Figure 8.5: A tableau which proves that the Quine-paradox is not satisfiable.



(8.34)

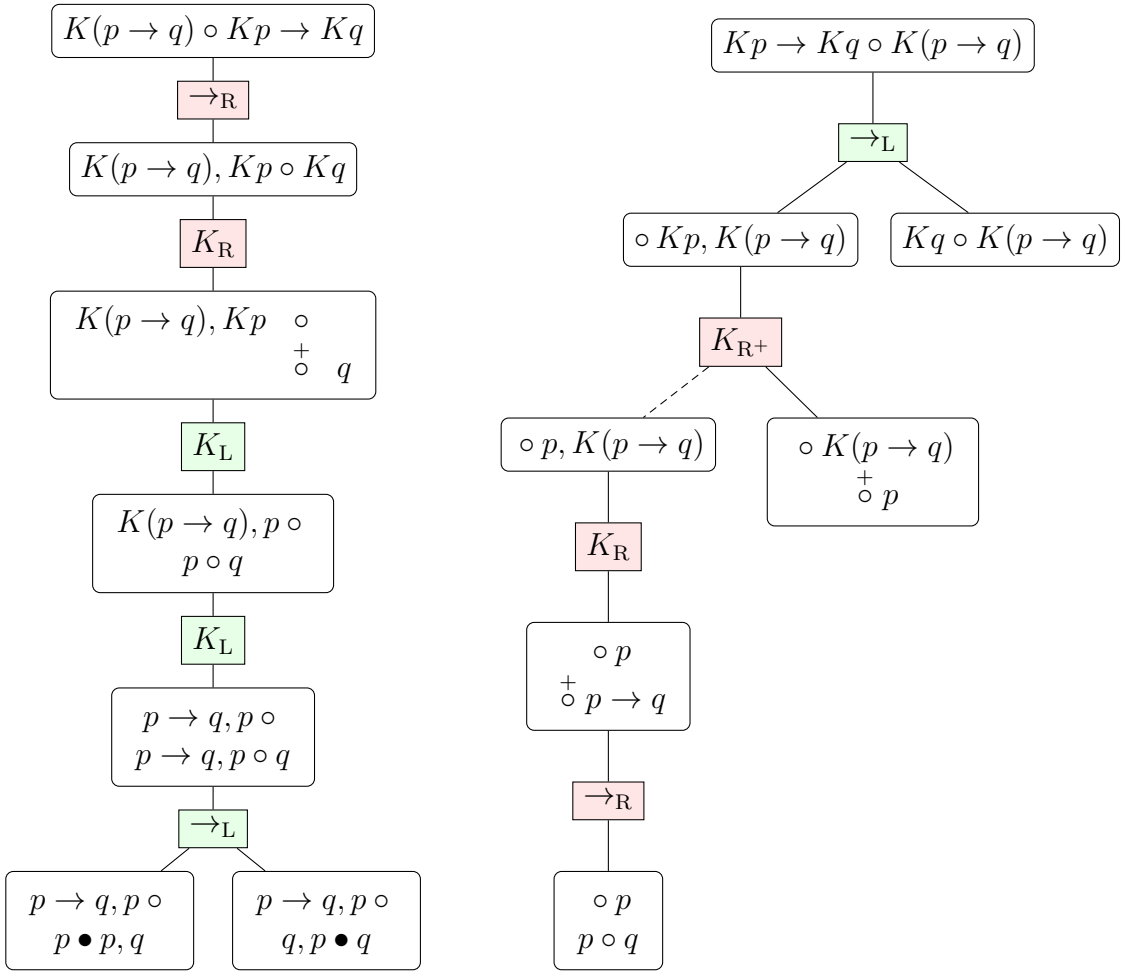
K_{L}: If a formula $K\varphi$ appears on the left part of at least one of the sequents in the box then remove this formula from those sequents and add φ to all the sequents in the box.

K_{R}: If a formula $K\varphi$ appears in the right part of at least one of the sequents then remove them and add the sequent $\circ\varphi$ to the box.

K_{R^+}: If a formula $K\varphi$ appears on the right part of at least one of the sequents then add the sequent $\circ\varphi$ to the box, *and* add a try-out branch with the original sequents of which one is extended with φ on the right part of it.

The symbol $\overset{+}{\circ}$ means that a new sequent (world) has been added to the box. This also implies that all formulas $K\varphi$ which were removed in preceding steps becomes active again. They are to be placed in left part of the first sequent of the sequent box.

Below two relatively simple examples are given. The first demonstrates that $K(p \to q) \models Kp \to Kq$ by means of a closed tableau. As you can see, the two end nodes consist of sequent boxes of which each contains a contradictory sequent. The second tableau shows that the converse of this inference is invalid: $Kp \to Kq \not\models K(p \to q)$.



(8.35)

Before we may jump to conclusions we need to be precise about closed and open multi-sequents. A multi-sequent is *closed* if it contains an impossible sequent containing contradictory information, i.e., a formula which appears on the left and on the right part of the sequent. All the worlds as described in a multi-sequent need to be possible to provide a counter-example. A tableau is closed if it contains only branches with closed multi-sequents in the terminal nodes. A multi-sequent is *open* if it is not closed and all of its sequents are free of logical symbols. A tableau is open if it contains at least one open multi-sequent. As for propositional and predicate logic, an open tableau detects invalidity and the open multi-sequent is nothing less than the description of a counter-model. The first sequent of the open node is then the world at which rejection of the inference takes place: all the premises are true there, and the conclusion will be false. A closed tableau tells us that the top sequent represents a valid inference.

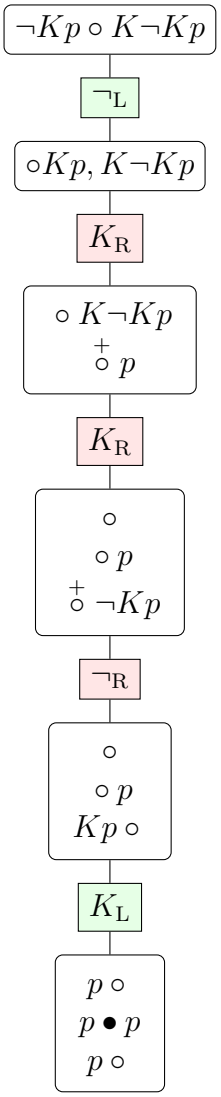
The first tableau in (8.35) showed a direct consequence of the logical closure property of the epistemic operator K, which holds for all 'necessity' operators in modal logics such as the dynamic operator $[\pi]$. The second tableau in (8.35) shows the invalidity of

the converse by means of the construction of a two worlds counter-model of which one falsifies p and the other verifies p and falsifies q. We have used the try-out version of K_{R} in the second step in order to find the smallest counter-model. The third step is a regular K_{R}-step because an additional try-out branch would close immediately ($p \bullet p, q$).

If we would have used K_{R} twice we would have end up with a three worlds counter-model. In other more complicated cases of invalid inference the try-out version of the K_{R} is really needed to find finite counter-models, just as we have seen for certain predicate logical inferences.

Exercise 8.15 Show with two tableaus that $Kp \vee Kq \models K(p \vee q)$ and $K(p \vee q) \not\models Kp \vee Kq$.

The following tableau shows a principle which holds specifically for epistemic logic: negative introspection.



(8.36)

The tableau ends with a description of a single ‘impossible’ possible worlds model. In

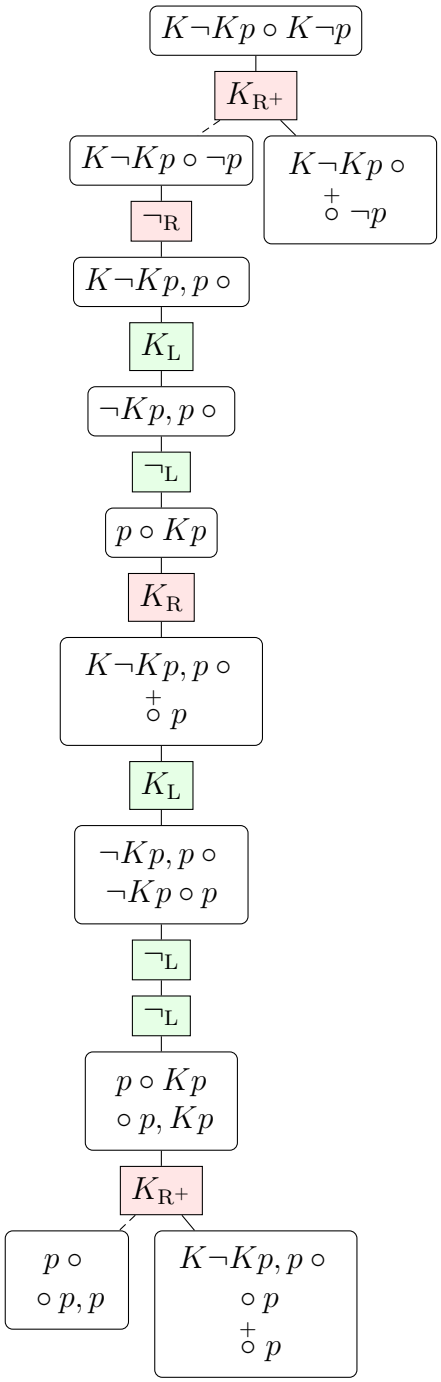
fact it tells us that a counter-model requires at least one impossible world at which p is both true and false, and therefore, a counter-model for negative introspection does not exist.

Exercise 8.16 Show with two tableaus that $Kp \models p$ and $p \not\models Kp$.

Exercise 8.17 Show with a closed tableau that $Kp \models KKp$ (positive introspection).

Exercise 8.18 Show with a closed tableau that $K(Kp \vee q) \models Kp \vee Kq$.

As a last example we demonstrate one other tableau where the try-out version of K_{R} are required to get a counter-model to compute an invalidity: $K\neg Kp \not\models K\neg p$. It says that if the agent knows that he does not know that p does not imply that he must know that $\neg p$ is the case. The tableau to find the smallest counter-model requires two applications of K_{R^+}.



(8.37)

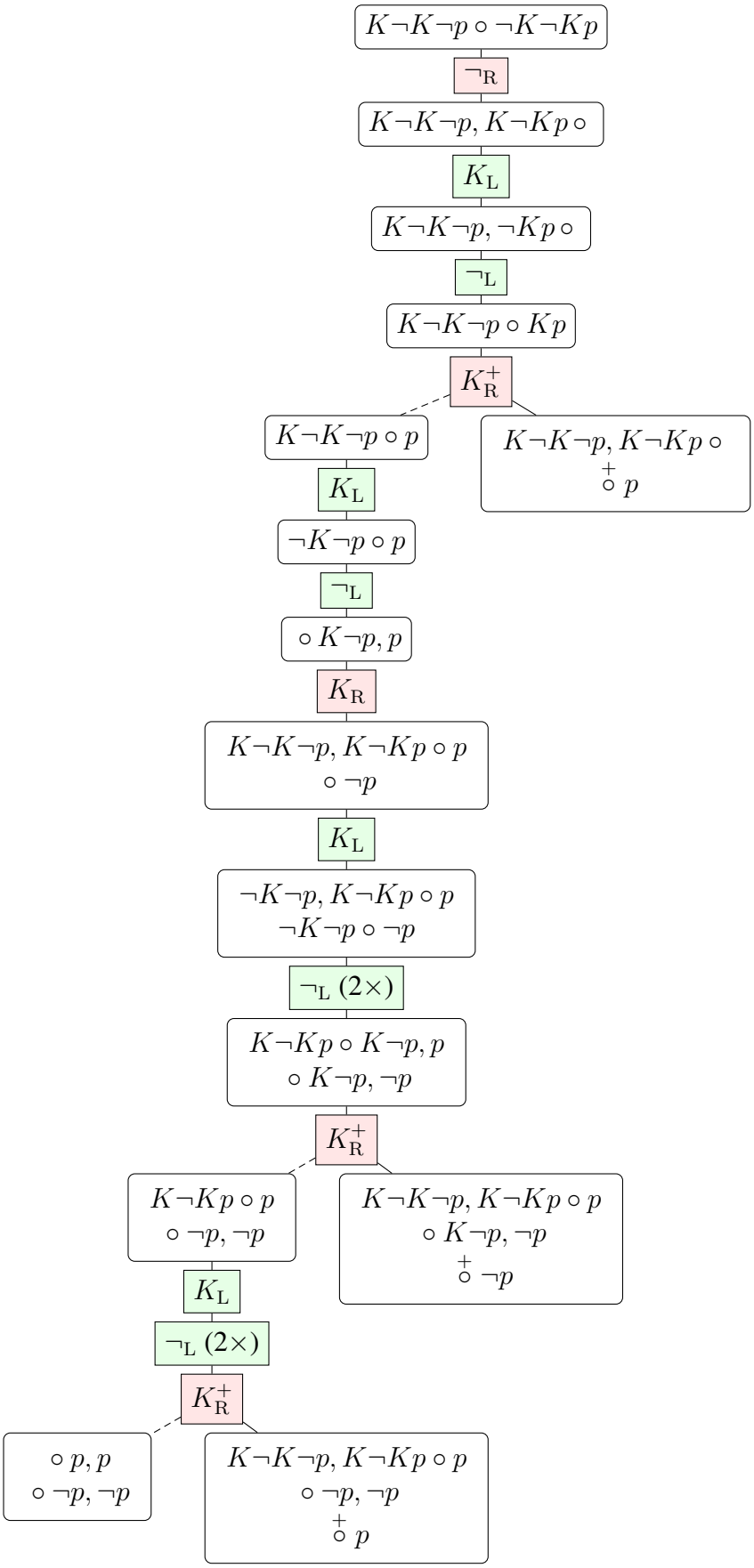
The counter-model which has been found in the left-most branch contains two worlds, one which verifies p and one which falsifies p. In both worlds the agent does not know that p and so $K\neg Kp$ is true in the first world (and also in the second), but $K\neg p$ is false in this world because p is false in the second.

Exercise 8.19 Show with a tableau that $K\neg K\neg p \not\models \neg K\neg Kp$.

Exercise 8.20 Show with a tableau that $\neg K \neg K p \models K \neg K \neg p$.

In many modal logics such as the epistemic and dynamic logic of the first part of this book the so-called *finite model property* holds. This means that there exist no inferences (with a finite set of premises) with only infinite counter-models such as we have seen for predicate logic in the example (8.31) on page 8-19. This also means that we can always detect invalidity for invalid inferences in single agent epistemic logic by using the tableau method with the given rules for the knowledge operator.

Single agent epistemic logic is by far the easiest modal logic when it comes down to defining a complete tableau system. For other modal logics this is much harder, but not impossible. Instead of multi-sequents so-called *hyper-sequents* are needed to search and specify counter-models by means of reduction rules. A hyper-sequent may not only contain multiple sequents but also other hyper-sequents. Using the format we have been using for single agent epistemic logic here this would look like nested boxes which can be used to capture the accessibility relation of potential counter-models. For multi-modal logics such as multi-agent epistemic logic and dynamic logic we need in addition labeling mechanisms for the nested boxes as to keep track of multiple accessibility relations. On top of that we also need quite complicated rules for 'path'-operators such as the common knowledge operator in multi-agent epistemic logic or the iteration operator in dynamic logic. All these technical complications are the main reason that tableau methods for advanced modal logics have not been standardized yet. Construction of models, whether they are realized by means of extended tableau techniques or alternative methods, are in the field of applied modal logic a very important theme of ongoing research. For sake of presentation and clarity, we do not want to drag along our readers into the highly technical mathematics of it.


$K\neg K\neg p \circ \neg K\neg Kp$
$\neg_{\mathrm{R}}$
$K\neg K\neg p, K\neg Kp \circ$
K_{L}
$K\neg K\neg p, \neg Kp \circ$
$\neg_{\mathrm{L}}$
$K\neg K\neg p \circ Kp$
K^{+}_{R}
$K\neg K\neg p \circ p$
$K\neg K\neg p, K\neg Kp \circ \overset{+}{\circ} p$
K_{L}
$\neg K\neg p \circ p$
$\neg_{\mathrm{L}}$
$\circ K\neg p, p$
K_{R}
$K\neg K\neg p, K\neg Kp \circ p \circ \neg p$
K_{L}
$\neg K\neg p, K\neg Kp \circ p \neg K\neg p \circ \neg p$
$\neg_{\mathrm{L}}\ (2\times)$
$K\neg Kp \circ K\neg p, p \circ K\neg p, \neg p$
K^{+}_{R}
$K\neg Kp \circ p \circ \neg p, \neg p$
$K\neg K\neg p, K\neg Kp \circ p \circ K\neg p, \neg p \overset{+}{\circ} \neg p$
K_{L}
$\neg_{\mathrm{L}}\ (2\times)$
K^{+}_{R}
$\circ p, p \circ \neg p, \neg p$
$K\neg K\neg p, K\neg Kp \circ p \circ \neg p, \neg p \overset{+}{\circ} p$

(8.38)

Chapter 9

Proofs

In the first part of this book we have discussed complete axiomatic systems for propositional and predicate logic. In the previous chapter we have introduced the tableau systems of Beth, which was a method to test validity. This method is much more convenient to work with since it tells you exactly what to do when a given formula has to be dealt with during such a validity test. Despite the convenience of Beth's system it does not represent the way humans argue.

In the late 1930s the German mathematician Gerhard Gentzen developed a system which he called *natural deduction*, in which the deduction steps as made in mathematical proofs are formalized. This system is based not so much on axioms but on rules instead. For each logical symbol, connectives and quantifiers, rules are given just in the way they are dealt with in mathematical proofs.

Gerhard Gentzen

Dag Prawitz

In this chapter we will demonstrate how this system works. The precise definition of

these rules goes back to the Swedish logician Dag Prawitz, who gave a very elegant reformulation of Gentzen's work in the 1960s.[1]

9.1 Natural deduction for propositional logic

In chapter 2 we have introduced an axiomatic system for propositional logic. By means of the rule of modus ponens one may jump from theorems to new theorems. In addition we had three axioms, theorems that do not have to be proven, and which may be used as starting points. Since these axioms are tautologies, and the rule of modus ponens is a sound rule of inference, a proof is then just a list of tautologies, propositions that are true under all circumstances.

Although this system is fun to work with for enthusiast readers who like combinatoric puzzles, it is surely not the way people argue. You may remember that it took us even five steps to prove that an extremely simple tautology as $\varphi \to \varphi$ is valid. This may even be worse for other trivial cases. It takes almost a full page to prove that $\varphi \to \neg\neg\varphi$ is valid (a real challenge for the fanatic puzzler)!

The pragmatic problem of a purely axiomatic system is that it does not facilitate a transparent manner of *conditional reasoning*, which makes it deviate very much from human argumentation. In an ordinary setting people derive conclusions which hold under *certain* circumstances, rather than summing up information which always hold. Especially when conditional propositions, such as the implicative formulas as mentioned here above, have to be proven the conditionals are used as presuppositions. Let us illustrate this with a simple mathematical example.

> If a square of a positive integer doubles the square of another positive integer then these two integers must both be even.
>
> Suppose m, n are two positive integers such that $m^2 = 2n^2$. This means m must be even, because if m^2 is even then m must be even as well. So, $m = 2k$ for some positive integer k. Since $m^2 = 2n^2$ we get $2n^2 = (2k)^2 = 4k^2$, and therefore, $n^2 = 2k^2$ which means that n must be even as well.

In the proof we presuppose that the antecedent ($m^2 = 2n^2$) of the conditional proposition ($m^2 = 2n^2 \to m, n$ even) that is to be proven holds. This is what is called an *hypothesis*. In the proof we derived that the consequent (m, n even) of the proposition holds under the circumstances that the hypothesis holds. The validity of this type of conditional reasoning reflects an important formal property of propositional logic (and also of the other logics

[1] The format of the proofs in this chapter has been introduced by the American logician John Fitch. Prawitz used tree like structures, whereas here, in analogy of Fitch's presentation proofs are divided into so-called subproofs.

which has been introduced in the first part of this book), which is called the *deduction property*. Formally it looks as follows: For every set of formulas Σ and for every pair of formulas φ and ψ:

$$\Sigma, \varphi \models \psi \text{ if and only if } \Sigma \models \varphi \rightarrow \psi \tag{9.1}$$

It says that by means of the implication we can reason about valid inference within the propositional language explicitly. A conditional proposition $\varphi \rightarrow \psi$ is a valid inference within a context Σ if and only if ψ is a valid conclusion from Σ extended with φ as an additional assumption (hypothesis). The deduction property reveals the operational nature of implication: φ leads to the conclusion ψ.

Exercise 9.1 Show that this deduction property holds for propositional logic by making use of truth tables.

Exercise 9.2 The modus ponens rule and the deduction property are characteristic for the implication in propositional logic. Let © be some propositional connective which has the modus ponens and deduction property:

$$\varphi, \varphi \copyright \psi \models \psi \qquad \varphi \models \psi \text{ if and only if } \models \varphi \copyright \psi$$

Show that © must be the implication $\rightarrow$.

Integration of the deduction property in a deduction system requires accommodation of hypotheses, i.e., additional assumptions that a reasoner uses in certain parts of his line of argumentation or proof. A proof of $\varphi \rightarrow \varphi$ then becomes trivial. Since assuming φ leads to φ we may conclude that $\varphi \rightarrow \varphi$ is always true. We may write this as follows:

$$\begin{array}{l} \left[\begin{array}{l} \underline{\varphi} \\ \hline \varphi \quad \text{repeat} \end{array}\right] \\ \varphi \rightarrow \varphi \quad \text{Ded} \end{array} \tag{9.2}$$

The first part between square brackets we call a *subproof* of the full proof. A subproof starts with an hypothesis (underlined) which is assumed to hold within this subproof. Proving a conditional proposition $\varphi \rightarrow \psi$ requires a subproof with hypothesis φ and conclusion ψ within this subproof. For our simple example (9.2) this immediately leads to success, but it may involve much more work for longer formulas. Consider the following example where we want to prove the second axiom of the axiomatic system as given in chapter 2: $(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))$.

$$\begin{array}{l} \left[\begin{array}{l} \underline{\varphi \rightarrow (\psi \rightarrow \chi)} \\ \hline \vdots \\ (\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi) \end{array}\right] \\ (\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi)) \quad \text{Ded} \end{array} \tag{9.3}$$

We have first set up a preliminary format of our proof. The conditional proposition that we want to prove has been rewritten as a subproof, which we have to establish later on. We need to show that the antecedent of the proposition indeed leads to the consequent. Since the desired conclusion of the subproof is an implication again we may follow the same procedure and extend our first format in the following way:

$$\begin{array}{l}
\left[\begin{array}{l}
\varphi \to (\psi \to \chi) \\
\hline
\left[\begin{array}{l}
\varphi \to \psi \\
\hline
\left[\begin{array}{l} \vdots \\ \varphi \to \chi \end{array}\right]
\end{array}\right] \\
(\varphi \to \psi) \to (\varphi \to \chi) \quad \text{Ded}
\end{array}\right] \\
(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)) \quad \text{Ded}
\end{array} \tag{9.4}$$

Here we have a subproof within a subproof, in which we need to show that the additional assumption $\varphi \to \psi$ leads to a conclusion $\varphi \to \chi$. This second hypothesis has been added to the hypothesis of the first subproof. In order to obtain the desired conclusion we may therefore use both hypotheses.

Again, the conclusion is a conditional proposition, and so, for the third time, we squeeze in a new subproof.

$$\begin{array}{l}
\left[\begin{array}{l}
\varphi \to (\psi \to \chi) \\
\hline
\left[\begin{array}{l}
\varphi \to \psi \\
\hline
\left[\begin{array}{l}
\varphi \\
\hline
\left[\begin{array}{l} \vdots \\ \chi \end{array}\right]
\end{array}\right] \\
\varphi \to \chi \quad \text{Ded}
\end{array}\right] \\
(\varphi \to \psi) \to (\varphi \to \chi) \quad \text{Ded}
\end{array}\right] \\
(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)) \quad \text{Ded}
\end{array} \tag{9.5}$$

Given this reformulation, we need to prove that χ holds given three hypotheses: $\varphi \to (\psi \to \chi)$, $\varphi \to \psi$ and φ. This is not very hard to prove by making use of our earlier rule of modus ponens. From the second and the third ψ follows and from the first and the third $\psi \to \chi$. These new propositional formulas can then be combined to establish χ. Here is

our final result:

$$\left[\begin{array}{l} \varphi \to (\psi \to \chi) \\ \hline \left[\begin{array}{l} \varphi \to \psi \\ \hline \left[\begin{array}{ll} \varphi & \\ \hline \psi & \text{MP} \\ \psi \to \chi & \text{MP} \\ \chi & \text{MP} \end{array}\right] \\ \varphi \to \chi \quad \text{Ded} \end{array}\right] \\ (\varphi \to \psi) \to (\varphi \to \chi) \quad \text{Ded} \end{array}\right] \tag{9.6}$$

$$(\varphi \to (\psi \to \chi)) \to ((\varphi \to \psi) \to (\varphi \to \chi)) \quad \text{Ded}$$

This result means that we no longer have to use the second axiom of the axiomatic system as described in chapter 2. It can be derived by means of our new deduction rule.

The first axiom of the system, $\varphi \to (\psi \to \varphi)$, can be established also quite straightforwardly by means of the deduction rule. In order to prove $\varphi \to (\psi \to \varphi)$ we need to show that $\psi \to \varphi$ can be proved from φ. This can be shown then by simply concluding that φ follows from φ and ψ:

$$\left[\begin{array}{l} \varphi \\ \hline \left[\begin{array}{ll} \psi & \\ \hline \varphi & \text{rep.} \end{array}\right] \\ \psi \to \varphi \quad \text{Ded} \end{array}\right] \tag{9.7}$$

$$\varphi \to (\psi \to \varphi) \quad \text{Ded}$$

Exercise 9.3 Prove $(\varphi \to (\psi \to \chi)) \to (\psi \to (\varphi \to \chi))$.

9.1.1 Proof by refutation

It seems that we can replace the axiomatic system by a natural deduction system by simply replacing the axioms by a single rule, the deduction rule. This is not the case, however. The third axiom of the axiomatic system $(\neg\varphi \to \neg\psi) \to (\psi \to \varphi)$, also called *contraposition*, can not be derived by deduction and modus ponens only. We need something to deal with the negations in this formula.

There seems to be a way out by taking $\neg\varphi$ to be an abbreviation of the conditional formula $\varphi \to \bot$. This establishes a procedure to prove negative information by means of the deduction rule. Proving $\neg\varphi$ requires a proof that the assumption that φ holds leads to a contradiction ($\bot$). This is indeed a natural way to establish negative information, as shown in the following example

$\sqrt{2}$ is not a rational number.

Suppose $\sqrt{2}$ were a rational number. This means there are two positive integers m and n such that $(m/n)^2 = 2$ and, in addition, that m or n is odd, since we can simply take the smallest pair such that $(m/n)^2 = 2$ (they cannot both be even since then it would not be the smallest pair for which this equation holds). But then $m^2 = 2n^2$ and therefore m and n must be even, as we have shown in an earlier example (page 9-2). Clearly, we have derived a contradiction, and therefore $\sqrt{2}$ must be an irrational number.

A reformulation in natural deduction style looks as follows:

$$\dfrac{\left[\begin{array}{l}\sqrt{2} \in \mathbb{Q} \\ \hline (m/n)^2 = 2 \text{ for certain pair of positive integers} \\ m, n \text{ with } m \text{ or } n \text{ being odd.} \\ m = 2n^2 \\ m \text{ and } n \text{ are both even positive integers} \\ \bot\end{array}\right]}{\neg(\sqrt{2} \in \mathbb{Q})} \tag{9.8}$$

This way of proving negative statements suffices to derive certain propositional logical theorems containing negative information. For example, the converse of the contraposition axiom can be established in this way;

$$\dfrac{\left[\begin{array}{l}\varphi \to \psi \\ \hline \left[\begin{array}{l}\psi \to \bot \\ \hline \left[\begin{array}{l}\varphi \\ \hline \psi \quad \text{MP} \\ \bot \quad \text{MP}\end{array}\right] \\ \varphi \to \bot \quad \text{Ded}\end{array}\right] \\ (\psi \to \bot) \to (\varphi \to \bot) \quad \text{Ded}\end{array}\right]}{(\varphi \to \psi) \to ((\psi \to \bot) \to (\varphi \to \bot)) \quad \text{Ded}} \tag{9.9}$$

Replacing $\to \bot$ by negations then settles $(\varphi \to \psi) \to (\neg\psi \to \neg\varphi)$. Unfortunately, this simple solution does not work for the axiom of contraposition. To get a complete system we need an additional rule.

Exercise 9.4 Show, by trying out the procedure which we have used for the previous examples, that you can not derive the axiom of contraposition by modus ponens and the deduction rule only.

This supplementary rule that we will need is in fact quite close to the deduction rule for negations. To derive a formula $\neg\varphi$ we prove that φ leads to a contradiction, which in fact says that φ can not be true. Our new rule says that φ can be proven by showing that φ can not be *false*. In terms of subproofs, if the hypothesis $\neg\varphi$ leads to a contradiction we may conclude that φ is the case. In this way we can prove the contraposition indeed.

$$
\begin{array}{|l}
\hline
1.\quad \neg\varphi \to \neg\psi \\
\hline
\quad \begin{array}{|ll}
\quad 2.\quad \psi & \\
\quad \begin{array}{|ll}
\quad 3.\quad \neg\varphi & \\
\hline
\quad 4.\quad \neg\psi & \text{MP 1,3} \\
\quad 5.\quad \bot & \text{MP 2,5}
\end{array} & \\
\quad 6.\quad \varphi & \text{new rule 3-5}
\end{array} \\
7.\quad \psi \to \varphi \qquad\qquad \text{Ded 2-6} \\
\hline
\end{array}
\qquad (9.10)
$$

$$
8.\quad (\neg\varphi \to \neg\psi) \to (\psi \to \varphi) \qquad \text{Ded 1-7}
$$

In step 6 we derived φ from the subproof $[\neg\varphi \mid \ldots \bot]$. The hypothesis that φ is false has led to a contradiction. The contradiction in 5 is obtained by a modus ponens, since $\neg\psi$ is an abbreviation of $\psi \to \bot$ here.

Exercise 9.5 Prove $((\varphi \to \psi) \to \varphi) \to \varphi$. Despite the absence of negations in this formula, you will need the new rule.

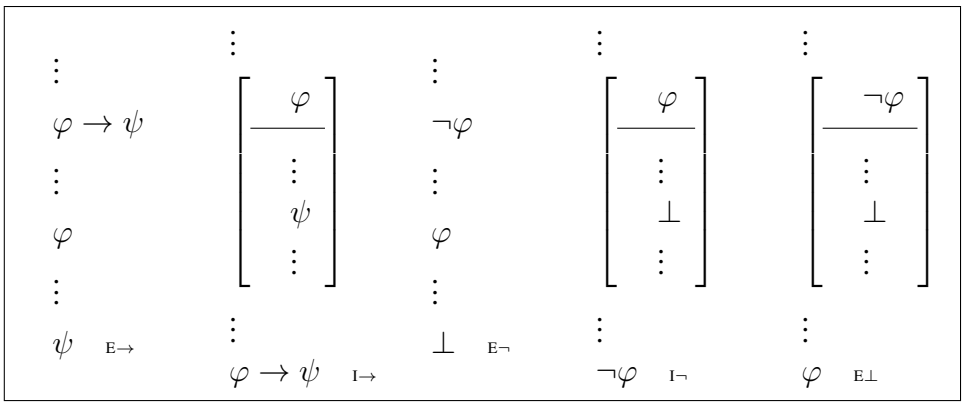
Exercise 9.6 Prove $\varphi \to \neg\neg\varphi$ and $\neg\neg\varphi \to \varphi$. Which of those proofs makes use of the new rule?

In logic this new rule is also called *proof by refutation*, or more academically, *reductio ad absurdum*. In fact, it captures the same way of reasoning as we have used in the tableau systems of the previous chapter. Proving the validity of an inference by presenting a closed tableau we show that the given formula can never be false, and therefore must be true, under the circumstances that the premises hold.

9.1.2 Introduction and elimination rules

The three rules suffice to obtain a complete system for propositional logic. The tradition in natural deduction is to separate the treatment of negations and implications which leads

to the following five rules.



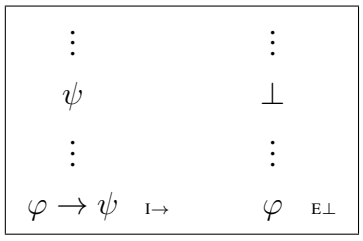
(9.11)

These rules are called *elimination* (E) and *introduction* (I) rules. The modus ponens is called an elimination rule since it says how to remove an implication $\varphi \to \psi$ and replace it by its consequent ψ. The rule then obtains the structural name E$\to$. Elimination of negation, E$\neg$, is then, as a consequence, the derivation of $\bot$ from $\neg\varphi$ and φ. The introduction rule for implication, I$\to$, is the deduction rule because it puts an implication on stage. I$\neg$ is defined analogously. The last rule represents the rule of proof by refutation and is most often seen as elimination of $\bot$ (E$\bot$).[2]

Two simpler versions of the deduction rule and the rule of proof by refutation are sometimes added to the system such that repetitions, as for example in the proof of $\varphi \to (\psi \to \varphi)$ as given in (9.7), can be avoided. If a statement φ is true then it also holds under arbitrary conditions: $\psi \to \varphi$. This is in fact a variation of the deduction rule (without hypothesis).

$$\dfrac{\dfrac{\varphi}{\psi \to \varphi}\ \text{I}\to\text{ 'simple'}}{\varphi \to (\psi \to \varphi)}\ \text{I}\to \tag{9.12}$$

For proofs of refutation the analogous simplification is called *ex falso*. Everything may be derived from a contradiction. We will use these simplified versions also in the sequel of this chapter. In general deductive form they look as follows:



(9.13)

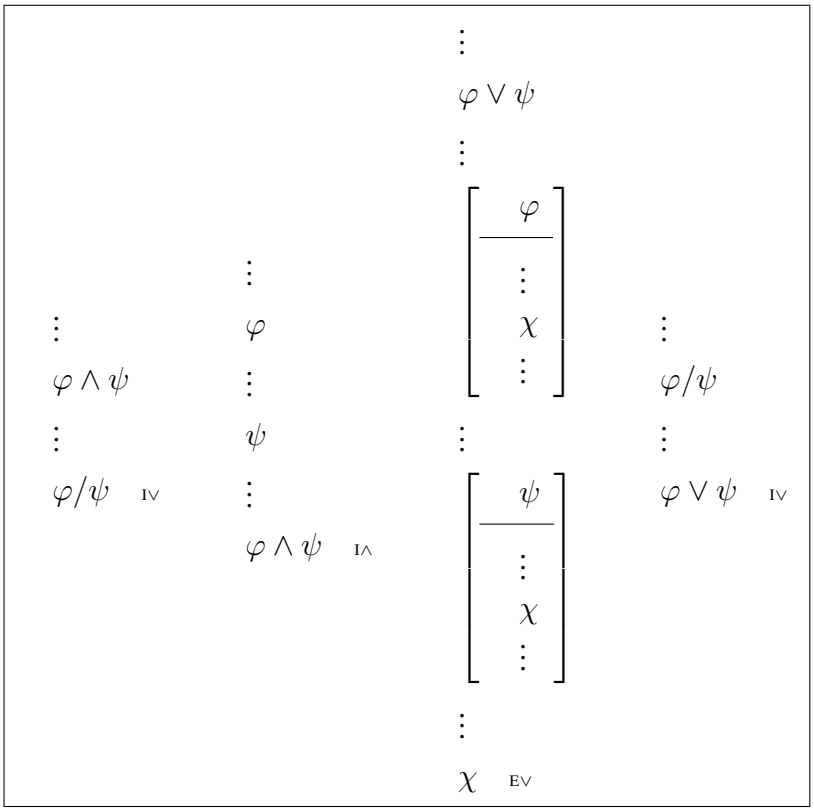
[2]Sometimes I$\neg$ is used for this rule, and then the introduction rule for negation is called falsum introduction (I$\bot$).

9.1.3 Rules for conjunction and disjunction

In propositional logic we also want to have rules for the other connectives. We could try the same procedure as we have done for negation. Find an equivalent formulation in terms of $\rightarrow$ and $\bot$ and then derive rules for these connectives.

$$\varphi \vee \psi \equiv (\varphi \rightarrow \bot) \rightarrow \psi \qquad \varphi \wedge \psi \equiv (\varphi \rightarrow (\psi \rightarrow \bot)) \rightarrow \bot \tag{9.14}$$

This option does not lead to what may be called a system of natural deduction. The equivalent conditional formulas are much too complicated. Instead, we use direct rules for manipulating conjunctive and disjunctive propositions. Below the introduction and elimination rules are given for the two connectives.



(9.15)

The rules for conjunction are quite straightforward. The elimination of a conjunction is carried out by selecting one of its arguments. Since we know that they are both true this is perfectly sound and a natural way of eliminating conjunctions. Introduction of a conjunction is just as easy. Derive a conjunction if both arguments have already been derived.

The introduction of a disjunction is also very simple. If you have derived one of the arguments then you may also derive the disjunction. The rule is perfectly correct but it is not very valuable, since in general, the disjunction contains less information then the information conveyed by one of the arguments.

The elimination of the disjunction is the most complicated rule. It uses two subproofs, one for each of the arguments of the disjunction. If in both subproofs, starting with one of the disjuncts (φ, ψ) as a hypothesis, the same information can be derived (χ) then we know that this must also hold in a context in which we are uncertain which of the arguments in fact holds $(\varphi \vee \psi)$. Despite the complexity of the rule, its soundness can be seen quite easily. We leave this to the reader in the next exercise.

Exercise 9.7 Show that $\Sigma, \varphi \vee \psi \models \chi$ if and only if $\Sigma, \varphi \models \chi$ and $\Sigma, \psi \models \chi$.

The elimination rule of disjunction reflects a natural way of dealing with uncertainty in argumentation. Here is an example of a mathematical proof.

> There exists two irrational numbers x and y such that x^y is rational.
>
> Let $z = \sqrt{2}^{\sqrt{2}}$. This number must be either irrational *or* rational. Although, we are uncertain about the status of z we can find in both cases two irrational x and y such that x^y must be rational.
>
> Suppose that z is rational, then we may take $x = y = \sqrt{2}$. We have just seen earlier that $\sqrt{2}$ is irrational, so this choice would be satisfactory.
>
> Suppose that z is irrational. Then we may take $x = z$ and $y = \sqrt{2}$, because then $x^y = z^{\sqrt{2}} = \sqrt{2}^{\sqrt{2}\cdot\sqrt{2}} = \sqrt{2}^2 = 2$, and that is a perfect rational number.

In the deduction style we could reformulate our argumentation as follows

$$
\begin{array}{l}
\sqrt{2}^{\sqrt{2}} \in \mathbb{Q} \vee \sqrt{2}^{\sqrt{2}} \notin \mathbb{Q} \\
\left[\begin{array}{l}
\sqrt{2}^{\sqrt{2}} \in \mathbb{Q} \\
\hline
x = y = \sqrt{2} \\
x^y = \sqrt{2}^{\sqrt{2}} \\
x^y \in \mathbb{Q} \text{ for certain } x, y \notin \mathbb{Q}
\end{array}\right] \\
\left[\begin{array}{l}
\sqrt{2}^{\sqrt{2}} \notin \mathbb{Q} \\
\hline
x = \sqrt{2}^{\sqrt{2}}, y = \sqrt{2} \\
x^y = 2 \\
x^y \in \mathbb{Q} \text{ for certain } x, y \notin \mathbb{Q}
\end{array}\right] \\
x^y \in \mathbb{Q} \text{ for certain } x, y \notin \mathbb{Q}
\end{array}
\tag{9.16}
$$

In practical reasoning disjunction elimination is also manifest as a way to jump to conclusions when only uncertain information is available. The following realistic scenario gives an illustration of this.

> I am traveling from A to B by train. If I run to the railway station of my home town A then I'll be in time to catch the train to B at 7.45AM, and then in B I will take the bus to the office and I will be there in time. If I won't run then I won't catch the 7.45AM train, but in this case I could take the train to B at 8.00AM instead. I would then need to take a cab from the railway station in B to arrive in time at the office. I start running to the railway station, not being sure whether my physical condition this morning will be enough to make me catch the first train (last night I have been to the cinema, and later on we went to the pub, etcetera). But no worries, I'll be in time at the office anyway (okay, it will cost me a bit more money if I won't catch the first train, since taking a cab is more expensive then taking the bus).

Here is the deductive representation of my reasoning:

$$\begin{array}{l}
\text{Catch 7.45AM-train} \ \vee \ \text{Catch 8.00AM-train} \\
\left[\begin{array}{l}\text{Catch 7.45AM-train.} \\ \hline \text{Take the bus in B to the office.} \\ \text{I'll be in time at the office.}\end{array}\right] \\
\left[\begin{array}{l}\text{Catch 8.00AM-train.} \\ \hline \text{Take a cab in B to the office.} \\ \text{I'll be in time at the office.}\end{array}\right] \\
\text{I'll be in time at the office.}
\end{array} \tag{9.17}$$

Exercise 9.8 Can you make up a similar scenario, jumping to safe conclusion while being uncertain about the conditions, from your personal daily experience? Now, reformulate this as a deduction such as given for the previous example.

Here is a very simple example of disjunction elimination in propositional logic. We derive $\psi \vee \varphi$ from the assumption $\varphi \vee \psi$:

$$\begin{array}{ll}
1. & \varphi \vee \psi \\
\left[\begin{array}{ll} 2. & \varphi \\ \hline 3. & \psi \vee \varphi \quad \text{I}\vee\ 2\end{array}\right] & \\
\left[\begin{array}{ll} 4. & \psi \\ \hline 5. & \psi \vee \varphi \quad \text{I}\vee\ 4\end{array}\right] & \\
6. & \psi \vee \varphi \quad \text{E}\vee\ 1,2\text{-}3,4\text{-}5
\end{array} \tag{9.18}$$

The formula $\psi \vee \varphi$ can be derived from φ and ψ by applying I$\vee$, so we can safely conclude $\psi \vee \varphi$ from $\varphi \vee \psi$ by E$\vee$.

Exercise 9.9 Prove $\varphi \to \psi$ from the assumption $\neg\varphi \vee \psi$.

Exercise 9.10 Prove $\neg(\varphi \wedge \psi)$ from $\neg\varphi \vee \neg\psi$.

Exercise 9.11 Prove $(\varphi \vee \psi) \wedge (\varphi \vee \chi)$ from $\varphi \vee (\psi \wedge \chi)$.

In general, disjunction elimination applies whenever we need to prove a certain formula χ from a disjunctive assumption $\varphi \vee \psi$. The strength of the elimination rule for disjunction is reflected by the equivalence of the inference $\varphi \vee \psi \models \chi$ on the one hand and the two inferences $\varphi \models \chi$ and $\psi \models \chi$ on the other (as you may have computed for yourself when you have made exercise 9.7 on page 9-10).

Disjunctive conclusions are much harder to establish in a deduction because of the earlier mentioned weakness of the introduction rule for disjunctions. Direct justification of a conclusion $\varphi \vee \psi$ by means of I$\vee$ requires a proof of one of the arguments, φ or ψ, which in many cases is simply impossible. Often a refutational proof is needed to obtain the desired disjunctive conclusion, that is, we show that $\neg(\varphi \vee \psi)$ in addition to the assumptions leads to a contradiction ($\bot$).

A clear illustration can be given by one of the most simple tautologies: $\varphi \vee \neg\varphi$. When it comes to reasoning with truth-values the principle simply says that there are only two opposite truth-values, and therefore it is also called 'principle of the excluded third' or 'tertium non datur'. From an operational or deductive point of view the truth of $\varphi \vee \neg\varphi$ is much harder to see. Since, in general, φ and $\neg\varphi$ are not tautologies, we have to prove that $\neg(\varphi \vee \neg\varphi)$ leads to a contradiction. Below a deduction, following the refutational strategy, has been given:

$$
\left[\begin{array}{lll}
1 & \neg(\varphi \vee \neg\varphi) & \\
\hline
 & \left[\begin{array}{lll} 2 & \varphi & \\ \hline 3 & \varphi \vee \neg\varphi & \text{I}\vee\ 2 \\ 4 & \bot & \text{E}\neg\ 1,3 \end{array}\right] & \\
5 & \neg\varphi & \text{I}\neg\ 2\text{-}4 \\
6 & \varphi \vee \neg\varphi & \text{I}\vee\ 5 \\
7 & \bot & \text{E}\to\ 1,6
\end{array}\right]
\qquad (9.19)
$$

$$
8 \quad \varphi \vee \neg\varphi \qquad \bot_{\text{E}}\ 1\text{-}7
$$

As you can see $\neg\varphi$ is derived from $\neg(\varphi \vee \neg\varphi)$ and this gives us finally the contradiction that we aimed at. In general this is the way to derive a disjunctive conclusion $\varphi \vee \psi$ for which a direct proof does not work. We assume the contrary $\neg(\varphi \vee \psi)$ then derive $\neg\varphi$ or $\neg\psi$ (or both) and show that this leads to a contradiction.

Exercise 9.12 Prove by a deduction that $(\varphi \to \psi) \vee (\psi \to \varphi)$ is a tautology.

Exercise 9.13 Deduce $\neg\varphi \vee \neg\psi$ from $\neg(\varphi \wedge \psi)$

Exercise 9.14 Prove by a deduction that $\neg\varphi \vee \psi$ follows from $\varphi \to \psi$.

9.2 Natural deduction for predicate logic

The natural deduction system for predicate logic consists of two simple rules and two more complicated, but at the same time more compelling, rules for the quantifiers $\forall$ and $\exists$. The easy weaker rules are $\forall$-elimination and $\exists$-introduction. They are just generalizations of the earlier elimination rule for $\wedge$ and the introduction rule for $\vee$.

From $\forall x\,\varphi$ we may derive that φ holds for 'everything'. This means that we substitute a term for x in φ. Substitution only has a small syntactic limitation. A term may contain variables, and we have to take care that no variable which occurs in such a 'substitute' gets bound by a quantifier in φ after replacing the occurrence of x by this term. If this is the case we say that this term is *substitutable* for x in φ. As an illustration that things go wrong when we neglect this limitation take the formula $\forall x \exists y\, \neg(x = y)$. Obviously, this formula is true in every model with more than one object. If we substitutes y for x in $\exists y\, \neg(x = y)$ we get $\exists y\, \neg(y = y)$ which is an inconsistent formula. The term y is not substitutable since y gets bound by the existential quantifier in $\exists y\, \neg(x = y)$.

Introduction of the existential quantifier works in the same way. If you have derived a property φ for certain term t you may replace this term by x and derive $\exists x\,\varphi$ successively. If we write $\varphi[t/x]$ for the result of substitution of t for x in φ and in addition prescribing that t must be substitutable for x in φ, we can formulate the rules mentioned here as follows:

$$\begin{array}{|ll|ll|}\hline \vdots & & \vdots & \\ \forall x\,\varphi & & \varphi[t/x] & \\ \vdots & & \vdots & \\ \varphi[t/x] & \text{E}\forall & \exists x\,\varphi & \text{I}\exists \\ \hline \end{array} \tag{9.20}$$

In practice these weak rules are only used to make small completing steps in a proof.

Also in the condition of I$\exists$ it is required that t is substitutable for x in φ. To see that neglecting this additional constraint leads to incorrect result take the formula $\forall y\, y = y$. This is a universally valid formula. It is also the result of replacing x by y in the formula $\forall y\, x = y$, but $\exists x \forall y\, x = y$ is certainly not a valid consequence of $\forall y\, y = y$: $\exists x \forall y\, x = y$ only holds in models containing only a single object.

The introduction rule of the universal quantifier is a bit more complicated rule, but, at the same time, it is a very strong rule. The rule is also referred at as *generalization*. By proving that a property φ holds for an *arbitrary* object we derive that φ holds for *all* objects: $\forall x\,\varphi$. To make sure that the object of which we prove φ-ness is indeed completely

arbitrary we use a new name which is not a constant in the language. Starting the subproof we extend the language with this new name only within the range of this subproof. Such an additional constant is also called a *parameter*. It may not be used in the main line of the proof which contains this subproof. Here is the formal version of the rule I$\forall$.

$$\begin{array}{|l}
\vdots \\
\left[\begin{array}{l} \qquad\qquad c \\ \hline \vdots \\ \varphi[c/x] \end{array}\right] \\
\vdots \\
\forall x\,\varphi \qquad\qquad \text{I}\forall
\end{array} \tag{9.21}$$

As you can see the subproof does not contain an hypothesis. The only information which is relevant here is that the parameter c does not appear in the line of the argument outside the subproof (represented by the vertical dots outside the subproof box), and that it is not a constant in the base language. To stress this minor syntactical limitation we indicate this c on top of the line where the subproof starts. This makes it clear that this is the reference to the arbitrary object for which we have to prove the desired property φ.

In natural settings of argumentation the generalization rule is most often combined with the deduction rule (I$\rightarrow$). If the assumption that an arbitrary object has the property φ leads to the conclusion that it also must have another property ψ we have proven that 'All φ are ψ' or in predicate logical notation $\forall x\,(\varphi \rightarrow \psi)$. In a formal deduction this looks as follows:

$$\begin{array}{l}
\left[\begin{array}{l} \qquad\qquad\qquad\qquad c \\ \hline \left[\begin{array}{l} \varphi[c/x] \\ \hline \vdots \\ \psi[c/x] \end{array}\right] \\ (\varphi \rightarrow \psi)[c/x] \qquad \text{I}\rightarrow \end{array}\right] \\
\forall x\,(\varphi \rightarrow \psi) \qquad \text{I}\forall
\end{array} \tag{9.22}$$

If we are able to prove for an arbitrary man that he must be mortal, we have proven that all men are mortal.

Take, as a mathematical example of this combination, the proof that if the square of a positive integer m doubles the square of another positive integer n, $m^2 = 2n^2$, they must both be even (page 9-2). The generalization rule, applied twice, would then rephrase this as universal result (given that the domain of discourse here contains only positive integers)

$$\forall x \forall y\,(x^2 = 2y^2 \rightarrow x, y \text{ both even})$$

Here is a first example deduction in predicate logic, showing that $\forall x\,(Px \wedge Qx)$ follows from two assumptions, $\forall x\, Px$ and $\forall x\, Qx$:

$$
\begin{array}{l}
1.\quad \forall x\, Px \qquad\qquad\qquad \text{Ass}\\
2.\quad \forall x\, Qx \qquad\qquad\qquad \text{Ass}\\
\left[\begin{array}{l}
3. \qquad\qquad\qquad\qquad c\\
\hline
4.\quad Pc \quad \text{E}\forall\ 1\\
5.\quad Qc \quad \text{E}\forall\ 2\\
6.\quad Pc \wedge Qc \quad \text{I}\wedge\ 4,5
\end{array}\right]\\
7.\quad \forall x\,(Px \wedge Qx) \quad \text{I}\forall\ 3\text{-}6
\end{array}
\tag{9.23}
$$

As you can see the generalization rule dominates the proof. It determines the external structure of the proof, whereas the weaker rule E∀ shows up only within the very inner part of the proof. The generalization rule works pretty much the same way as the deduction rule in propositional logic. In pure predicate logic a proof of a universal statement requires most often the generalization procedure. As we will see in the next section, there are other rules to prove statements with universal strength when we apply predicate logic for reasoning about a specific mathematical structure: the natural numbers.

Just as I∃ is a generalization of I∨, the elimination of existential quantifiers is taken care of by a generalization of disjunction elimination. A formula $\exists x\,\varphi$ represents that there is an object which has the property φ but, in general, we do not know who or what this φ-er is. To jump to a fixed conclusion we introduce an arbitrary φ-er, without caring about who or what this φ-er is, and show that this is enough to derive the conclusion that we are aiming at. The rule looks as follows:

$$
\boxed{
\begin{array}{l}
\vdots\\
\exists x\,\varphi\\
\vdots\\
\left[\begin{array}{l}
\varphi[c/x] \quad c\\
\hline
\vdots\\
\psi
\end{array}\right]\\
\vdots\\
\psi \qquad\qquad\qquad \text{E}\exists
\end{array}
}
\tag{9.24}
$$

The conclusion ψ can be derived on the basis of the introduction of an arbitrary φ-er (and nothing else). This means that if such a φ-er exists ($\exists x\,\varphi$) then we may safely conclude that ψ must hold. Again, the indication of the parameter c reminds us that it restricted by the same limitations as in the generalization rule I∀.

There is also a close relation with the generalization and the deduction rule. Combination of the latter two facilitated a way to prove statements of the form 'All φ are ψ'. In fact a slight variation of this is presupposed by means of the subproof in the E$\exists$- rule. Here it in fact says that it has been proven for an arbitrary object that if this object has the property φ then ψ must hold. And then we conclude that, given the assumption that there exists such a φ-er ($\exists x\, \varphi$), we know that ψ must hold.

The following variant of the train scenario as discussed on page 9-11 illustrates elimination of uncertainty conveyed by existential information in practice.

> Again, I am traveling from A to B. I don't know when trains leave, but I know at least there is a train departing from A going to B every half hour. Right now it is 7.35AM, and it will take me only ten minutes to get to the station. This means that I'll catch some train before 8.15AM: or *some point in time* t between 7.45AM and 8.15AM. The train from A to B takes 35 minutes, and my arrival at B will therefore be before 8.50AM (t + 35' $<$ 8.50AM). A cab ride will bring me in less than 15 minutes to the office and so I will be at the office before 9.05AM (t + 35' + 15' $<$ 9.05AM). This means I will be there before 9.15AM, when the first meeting of this morning starts.

Although I am quite uncertain about the time of departure I can safely conclude that I will be in time at the office.

Below a deduction is given which proves that $\forall x \exists y\, Rxy$ follows from $\exists y \forall x\, Rxy$. Each of the quantifier rules is used once:

$$
\begin{array}{lll}
1. & \exists y \forall x\, Rxy & \text{Ass} \\
\left[\begin{array}{lll}
2. & \forall x\, Rxc & c \\
\hline
\left[\begin{array}{lll} 3. & & d \end{array}\right] & & \\
\hline
4. & Rdc & \text{E}\forall\ 2 \\
5. & \exists y\, Rdy & \text{I}\exists\ 4
\end{array}\right] & & \\
6. & \forall x \exists y\, Rxy & \text{I}\forall\ 3\text{-}5 \\
7. & \forall x \exists y\, Rxy & \text{E}\exists\ 2\text{-}6
\end{array}
\tag{9.25}
$$

As an explanation what the individual steps in this proof mean, let us say that Rxy stands for 'x knows y' in some social setting. The assumption says there is some 'famous' person known by everybody. The conclusion that we want to derive means that 'everybody knows someone'. We started with E$\exists$, introducing an arbitrary person known by everybody, and we called him or her c ($\forall x\, Rxc$), and from this we want to derive the same conclusion ($\forall x \exists y\, Rxy$). To get this settled, we introduced an arbitrary object d and proved that d must know somebody ($\exists y\, Rdy$). This is proved by using Rdc (I$\exists$) which follows from $\forall x\, Rxc$ (E$\forall$).

Let us try a more difficult case: $\forall x\,Px \vee \exists x\,Qx$ follows from $\forall x\,(Px \vee Qx)$. The conclusion is a disjunction, and one can easily see that both disjuncts are not valid consequences of the given assumption. This means we have to prove this by using refutation (E$\bot$). We need to show that the assumption and the negation of the conclusion lead to a contradiction ($\bot$). Here is the complete deduction.

$$
\begin{array}{ll}
1. & \forall x\,(Px \vee Qx) \qquad \text{Ass} \\
\hline
\left[\begin{array}{ll}
2. & \neg(\forall x\,Px \vee \exists x\,Qx) \\
\hline
\left[\begin{array}{ll}
3. & c \\
\hline
4. & Pc \vee Qc \qquad \text{E}\forall\ 1 \\
\left[\begin{array}{ll} 5. & Pc \\ \hline \end{array}\right] & \\
\left[\begin{array}{ll}
6. & Qc \\
\hline
7. & \exists x\,Qx \qquad \text{I}\exists\ 6 \\
8. & \forall x\,Px \vee \exists x\,Qx \qquad \text{I}\exists\ 7 \\
9. & \bot \qquad \text{E}\neg\ 2,8 \\
10. & Pc \qquad \text{E}\bot\ 9 \text{ (simple version)}
\end{array}\right] & \\
11. & Pc \qquad \text{E}\vee\ 4,5\text{-}5,6\text{-}10
\end{array}\right] & \\
12. & \forall x\,Px \qquad \text{I}\forall\ 3\text{-}11 \\
13. & \forall x\,Px \vee \exists x\,Qx \qquad \text{I}\vee\ 12 \\
14. & \bot \qquad \text{E}\neg\ 2,13
\end{array}\right] & \\
15. & \forall x\,Px \vee \exists x\,Qx \qquad \text{E}\bot\ 2\text{-}14
\end{array}
\tag{9.26}
$$

In the outermost subproof we have shown that $\forall x\,(Px \vee Qx)$ in combination with $\neg(\forall x\,Px \vee \exists x\,Qx)$ leads to the conclusion $\forall x\,Px$ (12) which gives indeed an immediate contradiction (13,14). $\forall x\,Px$ can be obtained by proving the property P for an arbitrary object (c), which is carried in the second subproof. This can be proved then by using $Pc \vee Qc$ and disjunction elimination. Pc immediately follows from the first disjunct, Pc itself, and it also follows from Qc, since this leads to a contradiction and by applying ex falso, the simple form of proof by refutation, also to Pc.

Exercise 9.15 Prove that $\forall x\,\neg Px$ follows from $\neg\exists x\,Px$.

Exercise 9.16 Prove that $\exists x\,Px \wedge \exists x\,Qx$ follows from $\exists x\,(Px \wedge Qx)$.

Exercise 9.17 Prove that $\exists x\,\neg Px$ follows from $\neg\forall x\,Px$. You need to prove this by refutation, since a direct proof of the existential conclusion is not possible.

Exercise 9.18 Prove that $\exists x\,(Px \vee Qx)$ follows from $\exists x\,Px \vee \exists x\,Qx$, and also the other way around.

Exercise 9.19 (*) Prove that $\exists x\,(Px \to \forall x\,Px)$ is valid. This one requires a proof by refutation as well: show that $\bot$ follows from $\neg\exists x\,(Px \to \forall x\,Px)$.

9.2.1 Rules for identity

In addition to the rules for the quantifiers we also have to formulate rules for identity which are particularly important for mathematical proofs. The introduction rule is the simplest of all rules. It just states that an object is always equal to itself. It is in fact an axiom, there are no conditions which restrict application of this rule.

$$\begin{array}{l} \vdots \\ t = t \quad \text{I=} \end{array} \tag{9.27}$$

The elimination rule says that we always may replace terms by other terms which refer to the same object. We only have to take care that the variables which occur within these terms do not mess up the binding of variables by quantifiers. The term that we replace may only contain variables that occur freely (within the formula which is subject to the replacement), and the substitute may not contain variables which get bound after replacement. If these condition hold then we may apply the following rule:

$$\begin{array}{l} \vdots \\ t_1 = t_2 / t_2 = t_1 \\ \vdots \\ \varphi \\ \vdots \\ \varphi' \qquad \text{E=} \end{array} \tag{9.28}$$

where φ' is the result of replacing occurrences of t_1 by t_2 in φ (not necessarily all). Here are two simple examples showing the symmetry and transitivity of equality:

$$\begin{array}{llll} 1. & a = b & \text{Ass} & \\ 2. & a = a & \text{I=} & \\ 3. & b = a & \text{E= 1,2} & \end{array} \qquad \begin{array}{llll} 1. & a = b & \text{Ass} & \\ 2. & b = c & \text{Ass} & \\ 3. & a = c & \text{E= 1,2} & \end{array} \tag{9.29}$$

In the first derivation the first occurrence of a in 2 is replaced by b. In the second derivation the occurrence of b in 2 is replaced by a.

9.3 Natural deduction for natural numbers

In chapter 4 an axiomatic system of arithmetic, as introduced by the Italian logician and mathematician Giuseppe Peano, in predicate logical notation has been discussed.

Giuseppe Peano

In this section we want to give a natural deduction format for Peano's arithmetic, as an example of 'real' mathematical proof by means of natural deduction. These kind of systems are used for precise formalization of mathematical proofs, such that they can be checked, or sometimes be found (that is much harder of course), by computers.

Let us first repeat the axioms as discussed in chapter 4.

$$\begin{array}{ll} P1. & \forall x\,(x+0=x) \\ P2. & \forall x\,\forall y\,(x+sy=s(x+y)) \\ P3. & \forall x\,(x\cdot 0=0) \\ P4. & \forall x\,\forall y\,(x\cdot sy=x\cdot y+x) \\ P5. & \neg\exists x\,sx=0 \\ P6. & (\varphi[0/x]\wedge\forall x\,(\varphi\rightarrow\varphi[sx/x]))\rightarrow\forall x\,\varphi \end{array} \tag{9.30}$$

A straightforward manner to build a predicate logical system for arithmetic is to add these axioms to the system as has been introduced in the previous section. For the first five axioms we do not have an alternative. These axioms are then treated as rules without conditions, and can therefore be applied at any time at any place in a mathematical proof.

The last axiom, the principle of induction, can be reformulated as a conditional rule of deduction, in line with the way it is used in mathematical proofs. For the reader who is not familiar with the induction principle, the following simple example clarifies how it works.

> For every natural number n the sum of the first n odd numbers equals n^2.
>
> For 0 this property holds in a trivial way. The sum of the first zero odd numbers is an empty sum and therefore equals 0, which is also 0^2.
>
> Suppose that the property holds for a certain number k (induction hypothesis).
>
> $$1+3+\ldots+(2k-1)=k^2$$

We need to prove that under this condition the property must also hold for $k+1$ (sk). The sum of the first $k+1$ odd numbers is the same as

$$1 + 3 + ... + (2k - 1) + (2k + 1)$$

According the induction hypothesis, this must be equal to

$$k^2 + 2k + 1$$

and this equals $(k+1)^2$.

We have proven the property for 0 and also shown that if it holds for a certain natural number then it must also hold for its successors. From this we derive by induction that the property holds for all natural numbers.

9.3.1 The rule of induction

The inductive axiom in Peano's system can be rephrased as a conditional rule in the following way.

$$\begin{array}{l} \vdots \\ \varphi[0/x] \\ \vdots \\ \left[\begin{array}{l} \dfrac{\varphi[c/x] \quad {\scriptstyle c}}{\vdots} \\ \varphi[sc/x] \\ \vdots \end{array}\right] \\ \vdots \\ \forall x\, \varphi \qquad\qquad {\scriptstyle \text{Ind}} \end{array} \tag{9.31}$$

It mimics the format as has been described by the proof example here above. The formula $\varphi[0/x]$ says that the property φ holds for 0. The subproof represents the inductive step, and starts with the induction hypothesis. We assume $\varphi[c/x]$, i.e., an arbitrary φ-er represented by the parameter c (the induction hypothesis). If this assumption suffices to derive that the successor of c, sc, also must have the property φ, $\varphi[sc/x]$, then φ must hold for all objects, i.e., all the natural numbers.

In terms of the natural deduction system for predicate logic, the induction rule is an additional introduction rule for the universal quantifier. For some cases we can do without this rule and use the generalization rule instead. Here is a simple example which proves

that $x + 1$ coincides with the successor sx of x.

$$
\begin{array}{ll}
\left[\begin{array}{ll}
1. & c \\
\hline
2. & \forall x \forall y\,(x + sy = s(x + y)) \quad \text{P2} \\
3. & c + s0 = s(c + 0) \quad \text{E}\forall\ 2\ \text{(twice)} \\
4. & \forall x\,(x + 0 = x) \quad \text{P1} \\
5. & c + 0 = c \quad \text{E}\forall\ 4 \\
6. & c + s0 = sc \quad \text{E}=\ 5,3
\end{array}\right] & (9.32) \\
7. \quad \forall x\,(x + s0 = sx) \quad \text{I}\forall\ 1\text{-}6 &
\end{array}
$$

The proof demonstrates that this arithmetical theorem is a pure predicate logical consequence of the two first axioms of the Peano system.

In other cases we have to rely on the induction rule to derive a universal statement about the natural numbers. Here is a very simple example:

$$
\begin{array}{ll}
1. \quad 0 + 0 = 0 \quad \text{E}\forall\ \text{P1} & \\
\left[\begin{array}{ll}
2. & 0 + c = c \quad c \\
\hline
3. & 0 + sc = s(0 + c) \quad \text{E}\forall\ \text{P2 (twice)} \\
4. & 0 + sc = sc \quad \text{E}=\ 2,3
\end{array}\right] & (9.33) \\
5. \quad \forall x\,(0 + x = x) \quad \text{Ind}\ 1,2\text{-}4 &
\end{array}
$$

$0 + x = x$ is the property we have proved for all natural numbers x. First we have shown this is true for 0 and then in the induction step, the subproof 2-4, we have shown that the property $0 + c = c$ for an arbitrary c leads to $0 + sc = sc$.

Exercise 9.20 Prove that $\forall x\,(x \cdot s0 = x)$.

Exercise 9.21 Prove that $\forall x\,(0 \cdot x = 0)$.

The following proof uses both rules, generalization *and* induction.

$$
\left[\begin{array}{ll}
\hline
1. & c \\
\hline
2. & c+s0=sc \quad \text{E}\forall\ (9.32) \\
3. & sc+0=sc \quad \text{E}\forall\ \text{P1} \\
4. & c+s0=sc+0 \quad \text{E}=\ 3,2 \\
 & \left[\begin{array}{ll}
\hline
5. & c+sd=sc+d \qquad d \\
\hline
6. & c+ssd=s(c+sd) \quad \text{E}\forall\ \text{P2} \\
7. & c+ssd=s(sc+d) \quad \text{E}=\ 5,6 \\
8. & sc+sd=s(sc+d) \quad \text{E}\forall\ \text{P2} \\
9. & c+ssd=sc+sd \quad \text{E}=\ 8,7
\end{array}\right] \\
10. & \forall y\,(c+sy=sc+y) \quad \text{Ind } 4,5\text{-}9
\end{array}\right] \qquad (9.34)
$$

$$
11. \quad \forall x \forall y\,(x+sy=sx+y) \quad \text{I}\forall\ 1\text{-}10
$$

The outermost subproof justifies the use of a generalization in the last step, whereas the inner subproof contains the induction step of the inductive proof of 10.

Exercise 9.22 Prove that $\forall x\,\forall y\,(x+y=y+x)$ by filling in the gap represented by the vertical dots. You may use the theorem which has already been proved in (9.34).

$$
\left[\begin{array}{ll}
\hline
1. & c \\
\hline
 & \vdots \\
n. & c+0=0+c \quad \ldots \\
 & \left[\begin{array}{ll}
\hline
n+1. & c+d=d+c \qquad d \\
\hline
 & \vdots \\
n+k. & c+sd=sd+c \quad ..
\end{array}\right] \\
n+k+1. & \forall y\,(c+y=y+c) \quad \text{Ind } n,n+1\text{-}n+k
\end{array}\right]
$$

$$
n+k+2. \quad \forall x \forall y\,(x+y=y+x) \quad \text{I}\forall\ 1\text{-}n+k+1
$$

Exercise 9.23 Prove that $\forall x\,(x \cdot ss0 = x+x)$.

Exercise 9.24 (*) Prove that $\forall x \forall y\,(x \cdot y = y \cdot x)$.

Exercise 9.25 Prove that $\forall x \forall y \forall z\,(x+(y+z)=(x+y)+z)$.

9.4 Outlook

9.4.1 Completeness and incompleteness

9.4.2 Natural deduction, tableaus and sequents

9.4.3 Intuitionistic logic

9.4.4 Automated deduction

Chapter 10

Computation

Things You Will Learn in This Chapter This chapter gives a lightning introduction to computation with logic. First we will look at computing with propositional logic. You will learn how to put propositional formulas in a format suitable for computation, and how to use the so-called resolution rule. Next, we turn to computation with predicate logic. The procedure for putting predicate logical formulas into computational format is a bit more complicated. You will learn how to transform a predicate logical formula into a set of clauses. Next, in order to derive conclusions from predicate logical clauses, we need to apply a procedure called *unification*. Terms containing variables can sometimes made equal by means of substitution. We will present the so-called *unification algorithm*, and we will prove that if two terms can be made equal, then the unification algorithm computes the most general way of doing so. Finally, unification will be combined with resolution to give an inference mechanism that is very well suited for predicate logical computation, and we will see how this method is put to practical use in the Prolog programming language.

10.1 A Bit of History

Leibniz in his youth

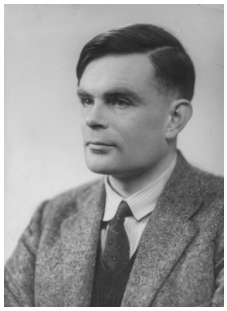
In 1673 the polymath Godfried Wilhelm Leibniz (1645–1716) demonstrated to the Royal

Society in London a design for a calculation device that was intended to solve mathematical problems by means of execution of logical inference steps. Leibniz was not only a mathematician, a philosopher and a historian, but also a diplomat, and he dreamed of rational approaches to conflict resolution. Instead of quarrelling without end or even resorting to violence, people in disagreement would simply sit down with their reasoning devices, following the adage *Calculemus* ("Let's compute the solution"). Mechanical computation devices were being constructed from that time on, and in 1928 the famous mathematician David Hilbert posed the challenge of finding a systematic method for mechanically settling mathematical questions formulated in a precise logical language.

David Hilbert

This challenge was called the *Entscheidungsproblem* ("the decision problem"). In 1936 and 1937 Alonzo Church and Alan Turing independently proved that it is impossible to decide algorithmically whether statements of simple school arithmetic are true or false. This result, now known as the Church-Turing theorem, made clear that a general solution to the *Entscheidungsproblem* is impossible. It follows from the Church-Turing theorem that a decision method for predicate logic does not exist. Still, it is possible to define procedures for computing inconsistency in predicate logic, provided that one accepts that these procedures may run forever for certain (consistent) input formulas.

Alonzo Church



Alan Turing

10.2 Processing Propositional Formulas

For computational processing of propositional logic formulas, it is convenient to first put them in a particular syntactic shape.

The simplest propositional formulas are called **literals**. A literal is a proposition letter or the negation of a proposition letter. Here is a BNF definition of literals. We assume that p ranges over a set of proposition letters P.

$$L \ ::= \ p \mid \neg p.$$

Next, a disjunction of literals is called a **clause**. Clauses are defined by the following BNF rule:

$$C \ ::= \ L \mid L \vee C.$$

Finally a CNF formula (formula in conjunctive normal form) is a conjunction of clauses. In a BNF rule:

$$\varphi \ ::= \ C \mid C \wedge \varphi.$$

Formulas in CNF are useful, because it is easy to test them for validity. For suppose φ is in CNF. Then φ consists of a conjunction $C_1 \wedge \cdots \wedge C_n$ of clauses. For φ to be valid, each conjunct clause C has to be valid, and for a clause C to be valid, it has to contain a proposition letter p and its negation $\neg p$. So to check φ for validity, find for each of its clauses C a proposition letter p such that p and $\neg p$ are both in C. In the next section, we will see that there is a simple powerful rule to check CNF formulas for satisfiability.

We will now start out from arbitrary propositional formulas, and show how to convert them into equivalent CNF formulas, in a number of steps. Here is the BNF definition of the language of propositional logic once more.

$$\varphi \ ::= \ p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Translating into CNF, first step The first step translates propositional formulas into equivalent formulas that are arrow-free: formulas without $\leftrightarrow$ and $\rightarrow$ operators. Here is how this works:

- Use the equivalence between $p \rightarrow q$ and $\neg p \vee q$ to get rid of $\rightarrow$ symbols.
- Use the equivalence of $p \leftrightarrow q$ and $(\neg p \vee q) \wedge (p \vee \neg q)$, to get rid of $\leftrightarrow$ symbols.

Here is the definition of arrow-free formulas of propositional logic:

$$\varphi \ ::= \ p \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi).$$

Translating into CNF, first step in pseudocode We will now write the above recipe in so-called *pseudocode*, i.e., as a kind of fake computer program. Pseudocode is meant to be readable by humans (like you), while on the other hand it is so close to computer digestible form that an experienced programmer can turn it into a real program as a matter of routine.

The pseudocode for turning a propositional formula into an equivalent arrow free formula takes the shape of a function. The function has a name, *ArrowFree*. A key feature of the definition of *ArrowFree* is that inside the definition, the function that is being defined is mentioned again. This is an example of a phenomenon that you will encounter often in recipes for computation. It is referred to as a *recursive function call*.

What do you have to do to make a formula of the form $\neg\psi$ arrow free? First you ask your dad to make ψ arrow free, and then you put $\neg$ in front of the result. The part where you ask your dad is the recursive function call.

function ArrowFree (φ):
/$\star$ precondition: φ is a formula. $\star$/
/$\star$ postcondition: ArrowFree (φ) returns arrow free version of φ $\star$/
begin function
case
  φ is a literal: **return** φ
  φ is $\neg\psi$: **return** $\neg$ ArrowFree (ψ)
  φ is $\psi_1 \wedge \psi_2$: **return** (ArrowFree (ψ_1) $\wedge$ ArrowFree (ψ_2))
  φ is $\psi_1 \vee \psi_2$: **return** (ArrowFree (ψ_1) $\vee$ ArrowFree (ψ_2))
  φ is $\psi_1 \rightarrow \psi_2$: **return** ArrowFree ($\neg\psi_1 \vee \psi_2$)
  φ is $\psi_1 \leftrightarrow \psi_2$: **return** ArrowFree $((\neg\psi_1 \vee \psi_2) \wedge (\psi_1 \vee \neg\psi_2))$
end case
end function

Note that the pseudocode uses comment lines: everything that is between /$\star$ and $\star$/ is a comment. The first comment of the function states the *precondition*. This is the assumption that the argument of the function is a propositional formula. This assumption is used in the function definition, for notice that the function definition follows the BNF definition of the formulas of propositional logic. The second comment of the function states the *postcondition*. This is the statement that all propositional formulas will be turned into equivalent arrow free formulas.

You can think of the precondition of a function recipe as a statement of rights, and of the postcondition as a statement of duties. The pre- and postcondition together form a *contract*: if the precondition is fulfilled (i.e., if the function is called in accordance with its rights) the function definition ensures that the postcondition will be fulfilled (the function will perform its duties). This way of thinking about programming is called *design by contract*.

Exercise 10.1 Work out the result of the function call ArrowFree $(p \leftrightarrow (q \leftrightarrow r))$.

Translating into CNF, second step Our next step is to turn an arrow free formula into a formula that only has negation signs in front of proposition letters. A formula in this shape is called a formula in *negation normal form*. Here is the BNF definition of formulas in negation normal form:

$$\begin{array}{lcl} L & ::= & p \mid \neg p \\ \varphi & ::= & L \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi). \end{array}$$

What this says is that formulas in negation normal form are formulas that are constructed out of literals by means of taking conjunctions and disjunctions.

The principles we use for translating formulas into negation normal form are the equivalence between$\neg(p \wedge q)$ and $\neg p \vee \neg q$, and that between $\neg(p \vee q)$ and $\neg p \wedge \neg q$. If we encounter a formula of the form $\neg(\psi_1 \wedge \psi_1)$, we can "push the negation sign inward" by replacing it with $\neg\psi_1 \vee \neg\psi_2$, and similarly for formulas of the form $\neg(\psi_1 \vee \psi_1)$. Again, we have to take care of the fact that the procedure will have to be carried out recursively. Also, if we encounter double negations, we can let them cancel out: formula $\neg\neg\psi$ is equivalent to ψ. Here is the pseudocode for turning arrow free formulas into equivalent formulas in negation normal form.

function NNF (φ):
/$\star$ precondition: φ is arrow-free. $\star$/
/$\star$ postcondition: NNF (φ) returns NNF of φ $\star$/
begin function
case
  φ is a literal: **return** φ
  φ is $\neg\neg\psi$: **return** NNF (ψ)
  φ is $\psi_1 \wedge \psi_2$: **return** (NNF (ψ_1) $\wedge$ NNF (ψ_2))
  φ is $\psi_1 \vee \psi_2$: **return** (NNF (ψ_1) $\vee$ NNF (ψ_2))
  φ is $\neg(\psi_1 \wedge \psi_2)$: **return** (NNF ($\neg\psi_1$) $\vee$ NNF ($\neg\psi_2$))
  φ is $\neg(\psi_1 \vee \psi_2)$: **return** (NNF ($\neg\psi_1$) $\wedge$ NNF ($\neg\psi_2$))
end case
end function

Again, notice the recursive function calls. Also notice that there is a contract consisting of a precondition stating that the input to the NNF function has to be arrow free, and guaranteeing that the output of the function is an equivalent formula in negation normal form.

Exercise 10.2 Work out the result of the function call NNF $(\neg(p \vee \neg(q \wedge r)))$.

Translating into CNF, third step The third and final step takes a formula in negation normal form and produces an equivalent formula in conjunctive normal form. This function uses an auxiliary function DIST, to be defined below. Intuitively, DIST(ψ_1, ψ_2) gives the CNF of the *disjunction* of ψ_1 and ψ_2, on condition that ψ_1, ψ_2 are themselves in CNF.

function CNF (φ):
/$\star$ precondition: φ is arrow-free and in NNF. $\star$/
/$\star$ postcondition: CNF (φ) returns CNF of φ $\star$/
begin function
case
  φ is a literal: **return** φ
  φ is $\psi_1 \wedge \psi_2$: **return** CNF (ψ_1) $\wedge$ CNF (ψ_2)
  φ is $\psi_1 \vee \psi_2$: **return** DIST (CNF (ψ_1), CNF (ψ_2))
end case
end function

Translating into CNF, auxiliary step The final thing that remains is define the CNF of the disjunction of two formulas φ_1, φ_2 that are both in CNF. For that, we use:

- $(p \wedge q) \vee r$ is equivalent to $(p \vee r) \wedge (q \vee r)$,
- $p \vee (q \wedge r)$ is equivalent to $(p \vee q) \wedge (p \vee r)$.

The assumption that φ_1 and φ_2 are themselves in CNF helps us to use these principles. The fact that φ_1 is in CNF means that either φ_1 is a conjunction $\psi_{11} \wedge \psi_{12}$ of clauses, or it is a single clause. Similarly for φ_2. This means that either at least one of the two principles above can be employed, or both of φ_1, φ_2 are single clauses. In this final case, $\varphi_1 \vee \varphi_2$ is in CNF.

function DIST (φ_1, φ_2):
/$\star$ precondition: φ_1, φ_2 are in CNF. $\star$/
/$\star$ postcondition: DIST (φ_1, φ_2) returns CNF of $\varphi_1 \vee \varphi_2$ $\star$/
begin function
case
  φ_1 is $\psi_{11} \wedge \psi_{12}$: **return** DIST ($\psi_{11}$,$\varphi_2$) $\wedge$ DIST (ψ_{12},φ_2)
  φ_2 is $\psi_{21} \wedge \psi_{22}$: **return** DIST ($\varphi_1$,$\psi_{21}$) $\wedge$ DIST (φ_1,ψ_{22})
  otherwise: **return** $\varphi_1 \vee \varphi_2$
end case
end function

In order to put a propositional formula φ in conjunctive normal form we can proceed as follows:

(1) First remove the arrows $\rightarrow$ and $\leftrightarrow$ by means of a call to ArrowFree.

(2) Next put the result of the first step in negation normal form by means of a call to NNF.

(3) Finally, put the result of the second step in conjunctive normal form by means of a call to CNF.

In other words, if φ is an arbitrary propositional formula, then

$$\text{CNF}(\text{NNF}(\text{ArrowFree}(\varphi)))$$

gives an equivalent formula in conjunctive normal form.

Exercise 10.3 Work out the result of the function call CNF $((p \vee \neg q) \wedge (q \vee r))$.

Exercise 10.4 Work out the result of the function call CNF $((p \wedge q) \vee (p \wedge r) \vee (q \wedge r))$.

10.3 Resolution

It is not hard to see that if $\neg\varphi \vee \psi$ is true, and $\varphi \vee \chi$ is also true, then $\psi \vee \chi$ has to be true as well. For assume $\neg\varphi \vee \psi$ and $\varphi \vee \chi$ are true. If φ is true, then it follows from $\neg\varphi \vee \psi$ that ψ. If on the other hand $\neg\varphi$ is true, then it follows from $\varphi \vee \chi$ that χ. So in any case we have $\psi \vee \chi$. This inference principle is called *resolution*. We can write the resolution rule as:

$$\frac{\neg\varphi \vee \psi \quad \varphi \vee \chi}{\psi \vee \chi}$$

Note that Modus Ponens can be viewed as a special case of this. Modus Ponens is the rule:

$$\frac{\varphi \to \psi \quad \varphi}{\psi}$$

But this can be written with negation and disjunction:

$$\frac{\neg\varphi \vee \psi \quad \varphi \vee \bot}{\psi}$$

The idea of resolution leads to a powerful inference rule if we apply it to two clauses. Clauses are disjunctions of literals, so suppose have two clauses $A_1 \vee \cdots \vee A_n$ and $B_1 \vee \cdots \vee B_m$, where all of the A and all of the B are literals. Assume that A_i and B_j are complements (one is the negation of the other, i.e., one has the form p and the other the form $\neg p$). Then the following inference step is valid:

$$\frac{A_1 \vee \cdots \vee A_n \qquad B_1 \vee \cdots \vee B_m}{A_1 \vee \cdots \vee A_{i-1} \vee A_{i+1} \vee \cdots \vee A_n \vee B_1 \vee \cdots \vee B_{j-1} \vee B_{j+1} \vee \cdots \vee B_m}$$

This rule is called the *resolution rule*. It was proposed by J. Alan Robinson (one of the inventors of the Prolog programming language) in 1965, in a landmark paper called "A Machine-Oriented Logic Based on the Resolution Principle." The rule allows to fuse two clauses together in a single clause.

Before we go on, it is convenient to switch to set notation. Let us say that a clause is a *set of literals*, and a *clause form* a *set of clauses*. Then here is an example of a clause form:

$$\{\{p, \neg q, r, \neg r\}, \{p, \neg p\}\}.$$

Resolution can now be described as an operation on pairs of clauses, as follows:

$$\frac{C_1 \cup \{p\} \qquad \{\neg p\} \cup C_2}{C_1 \cup C_2}$$

Alternatively, we may view resolution as an operation on clause forms, as follows:

$$\frac{C_1, \ldots, C_i \cup \{p\}, \{\neg p\} \cup C_{i+1}, C_{i+2}, \ldots, C_n}{C_1, \ldots, C_i \cup C_{i+1}, C_{i+2}, \ldots, C_n}$$

The empty clause, notation $[]$, corresponds to an empty disjunction. To make a disjunction true, at least one of the disjuncts has to be true. It follows that the empty clause is always *false*.

The empty clause form, notation $\emptyset$, corresponds to an empty conjunction, for clause form is conjunctive normal form. A conjunction is true if all of its conjuncts are true. It follows that the empty clause form is always *true*.

Exercise 10.5 Suppose a clause C_i contains both p and $\neg p$, for some proposition letter p. Show that the following rule can be used to simplify clause forms:

$$\frac{C_1, \ldots, C_i, \ldots, C_n}{C_1, \ldots, C_{i-1}, C_{i+1}, \ldots, C_n} \; p \in C_i, \neg p \in C_i$$

You have to show that this rule is *sound*. Assuming that the premise is true, show that the conclusion is also true.

If a clause form has $[]$ (the empty clause) as a member, then, since $[]$ is always false, and since clause forms express conjunctions, the clause form is always *false*. In other words, a clause form that has $[]$ as a member expresses a contradiction. So if we can derive the empty clause $[]$ from a clause form, we know that the clause form is *not satisfiable*.

Thus, resolution can be used as a *refutation technique*. To check whether ψ follows logically from $\varphi_1, \ldots, \varphi_n$, check whether the clause form corresponding to

$$\varphi_1 \wedge \cdots \wedge \varphi_n \wedge \neg\psi$$

is satisfiable, by attempting to derive the empty clause [] from the clause form, by means of the resolution rule. If the clause form is not satisfiable, the original inference is valid.

Example: we want to check whether from $\neg p \vee \neg q \vee r$, and $\neg p \vee q$ it follows that $\neg p \vee r$. Construct the formula

$$(\neg p \vee \neg q \vee r) \wedge (\neg p \vee q) \wedge \neg(\neg p \vee r).$$

This is the conjunction of the premisses together with a negation of the conclusion. Bring this in conjunctive normal form:

$$(\neg p \vee \neg q \vee r) \wedge (\neg p \vee q) \wedge p \wedge \neg r.$$

Write this formula in clause form:

$$\{\{\neg p, \neg q, r\}, \{\neg p, q\}, \{p\}, \{\neg r\}\}.$$

Applying resolution for $\neg q, q$ to the first two clauses gives:

$$\{\{\neg p, r\}, \{p\}, \{\neg r\}\}.$$

Applying resolution for $\neg p, p$ to the first two clauses gives:

$$\{\{r\}, \{\neg r\}\}.$$

Applying resolution for $r, \neg r$ gives:

$$\{[]\}$$

We have derived a clause form containing the empty clause. This is a proof by resolution that the inference is valid. We have tried to construct a situation where the premisses are true and the conclusion is false, but this attempt has led us to a contradiction. No doubt you will have noticed that this *refutation strategy* is quite similar to the strategy behind tableau style theorem proving.

Exercise 10.6 Test the validity of the following inferences using resolution:

(1) $((p \vee q) \wedge \neg q) \rightarrow r, q \leftrightarrow \neg p \models r$

(2) $(p \vee q) \rightarrow r, \neg q, \neg q \leftrightarrow p \models r$

Exercise 10.7 Determine which of the following clause forms are satisfiable:

(1) $\{\{\neg p, q\}, \{\neg q\}, \{p, \neg r\}, \{\neg s\}, \{\neg t, s\}, \{t, r\}\}$

(2) $\{\{p, \neg q, r\}, \{q, r\}, \{q\}, \{\neg r, q\}, \{\neg p, r\}\}$

Exercise 10.8 You are a professor and you are trying to organize a congress. In your attempt to draw up a list of invited speakers, you are considering professors a, b, c, d, e, f. Unfortunately, your colleagues have big egos, and informal consultation concerning their attitudes towards accepting your invitation reveals the following constraints:

- At least one of a, b is willing to accept.
- Exactly two of a, e, f will accept.
- b will accept if and only if c also accepts an invitation.
- a will accept if and only if d will not get invited.
- Similarly for c and d.
- If d will not get an invitation, e will refuse to come.

Use propositional logic to set up a clause set representing these constraints. (Hint: first express the constraints as propositional formulas, using proposition letters a, b, c, d, e, f. Next, convert this into a clause form.)

Exercise 10.9 As it turns out, there is only one way to satisfy all constraints of Exercise 10.8. Give the corresponding propositional valuation. (Hint: you can use resolution to simplify the clause form of the previous exercise.)

We know that checking (un)satisfiability for propositional logic can always be done. It cannot always be done efficiently. The challenge of building so called *sat solvers* for propositional logic is to speed up satisfiability checking for larger and larger classes of propositional formulas. Modern sat solvers can check satisfiability of clause forms containing hundreds of proposition letters. The usual way to represent a clause form is as a list of lines of integers. Here is an example of this so-called DIMACS format:

```
c Here is a comment.
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

The first line gives a comment (that's what it says, and what is says is correct). The second line states that this is a problem in conjunctive normal form with five proposition letters and three clauses. Each of the next three lines is a clause. 0 indicates the end of a clause. The home page of a popular sat solver called *MiniSat* can be found at `http://minisat.se/`. MiniSat calls itself a minimalistic, open-source SAT solver. It was developed to help researchers and developers to get started on SAT. So this is where *you* should start also if you want to learn more. Running the example (stored in file `sat.txt`) in *MiniSat* gives:

```
jve@vuur:~/tmp$ minisat2 sat.txt
This is MiniSat 2.0 beta
WARNING: for repeatability, setting FPU to use double precision
============================[ Problem Statistics ]=============================
|                                                                             |
|  Number of variables:  5                                                    |
|  Number of clauses:    3                                                    |
|  Parsing time:         0.00         s                                       |
============================[ Search Statistics ]==============================
| Conflicts |          ORIGINAL         |          LEARNT          | Progress |
|           |    Vars  Clauses Literals |    Limit  Clauses Lit/Cl |          |
===============================================================================
|         0 |       0        0        0 |        0        0    nan |  0.000 % |
===============================================================================
restarts              : 1
conflicts             : 0              (nan /sec)
decisions             : 1              (0.00 % random) (inf /sec)
propagations          : 0              (nan /sec)
conflict literals     : 0              ( nan % deleted)
Memory used           : 14.58 MB
CPU time              : 0 s

SATISFIABLE
```

Now let's have another look at the earlier clause form we computed:

$$\{\{\neg p, \neg q, r\}, \{\neg p, q\}, \{p\}, \{\neg r\}\}.$$

Written with indices, it looks like this:

$$\{\{\neg p_1, \neg p_2, p_3\}, \{\neg p_1, p_2\}, \{p_1\}, \{\neg p_3\}\}.$$

And here is the clause form in DIMACS format:

```
p cnf 4 3
-1 -2 3 0
-1 2 0
1 0
-3 0
```

If this text is stored in file `sat2.txt` then here is the result of feeding it to `minisat`:

```
jve@vuur:~/tmp$ minisat2 sat2.txt
This is MiniSat 2.0 beta
WARNING: for repeatability, setting FPU to use double precision
============================[ Problem Statistics ]=============================
|                                                                             |
|  Number of variables:  4                                                    |
|  Number of clauses:    3                                                    |
|  Parsing time:         0.00         s                                       |
Solved by unit propagation
UNSATISFIABLE
```

General background on propositional satisfiability checking can be found at `http://www.satisfiability.org/`.

10.4 Automating Predicate Logic

Alloy (`http://alloy.mit.edu`) is a software specification tool based on first order logic plus some relational operators. Alloy automates predicate logic by using bounded exhaustive search for counterexamples in small domains [Jac00]. Alloy does allow for automated checking of specifications, but only for small domains. The assumption that most software design errors show up in small domains is known as the *small domain hypothesis* [Jac06]. The *Alloy* website links to a useful tutorial, where the three key aspects of Alloy are discussed: logic, language and analysis.

The *logic* behind Alloy is predicate logic plus an operation to compute the transitive closures of relations. The transitive closure of a relation R is by definition the smallest transitive relation that contains R.

Exercise 10.10 Give the transitive closures of the following relations. (Note: if a relation is already transitive, the transitive closure of a relation is that relation itself.)

(1) $\{(1,2),(2,3),(3,4)\}$,

(2) $\{(1,2),(2,3),(3,4),(1,3),(2,4)\}$,

(3) $\{(1,2),(2,3),(3,4),(1,3),(2,4),(1,4)\}$,

(4) $\{(1,2),(2,1)\}$,

(5) $\{(1,1),(2,2)\}$.

The language is the set of syntactic conventions for writing specifications with logic. The analysis of the specifications takes place by means of bounded exhaustive search for counterexamples. The technique used for this is translation to a propositional satisfiability problem, for a given domain size.

Here is an example of a check of a fact about relations. We just defined the transitive closure of a relation. In a similar way, the symmetric closure of a relation can be defined. The symmetric closure of a relation R is the smallest symmetric relation that contains R.

We call the *converse* of a binary R the relation that results from changing the direction of the relation. A common notation for this is $R^\smile$. The following holds by definition:

$$R^\smile \;=\; \{(y,x) \mid (x,y) \in R\}.$$

We claim that $R \cup R^\smile$ is the symmetric closure of R. To establish this claim, we have to show two things: (i) $R \cup R^\smile$ is symmetric, and (ii) $R \cup R^\smile$ is the least symmetric relation that contains R. (i) is obvious. To establish (ii), we assume that there is some symmetric relation S with $R \subseteq S$ (S contains R). If we can show that $R \cup R^\smile$ is contained in S we know that $R \cup R^\smile$ is the least relation that is symmetric and contains R, so that it has to be the symmetric closure of R, by definition.

So assume $R \subseteq S$ and assume S is symmetric. Let $(x, y) \in R \cup R^{\smile}$. We have to show that $(x, y) \in S$. From $(x, y) \in R \cup R^{\smile}$ it follows either that $(x, y) \in R$ or that $(x, y) \in R^{\smile}$. In the first case, $(x, y) \in S$ by $R \subseteq S$, and we are done. In the second case, $(y, x) \in R$, and therefore $(y, x) \in S$ by $R \subseteq S$. Using the fact that S is symmetric we see that also in this case $(x, y) \in S$. This settles $R \cup R^{\smile} \subseteq S$.

Now that we know what the symmetric closure of R looks like, we can define it in predicate logic, as follows:

$$Rxy \vee Ryx.$$

Now here is a question about operations on relations. Given a relation R, do the following two procedures boil down to the same thing?

First take the symmetric closure, next the transitive closure

First take the transitive closure, next the symmetric closure

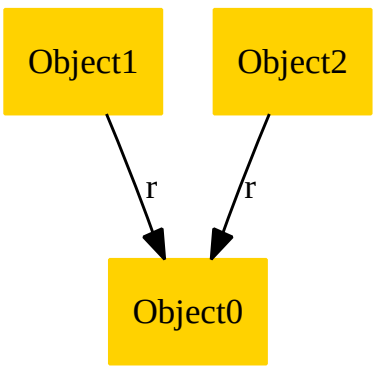
If we use R^+ for the transitive closure of R and $R \cup R^{\smile}$ for the symmetric closure, then the question becomes:

$$(R \cup R^{\smile})^+ \stackrel{?}{=} R^+ \cup R^{+\smile}$$

Here is an Alloy version of this question:

```
sig Object { r : set Object }
assert claim { *(r + ~r) = *r + ~*r }
check claim
```

If you run this in Alloy, the system will try to find counterexamples. Here is a counterexample that it finds:



To see that this is indeed a counterexample, note that for this R we have:

$$\begin{aligned}
R &= \{(1,0),(2,0)\} \\
R^{\smile} &= \{(0,1),(0,2)\} \\
R \cup R^{\smile} &= \{(1,0),(2,0),(0,1),(0,2)\} \\
R^{+} &= \{(1,0),(2,0)\} \\
R^{\smile +} &= \{(0,1),(0,2)\} \\
R^{+} \cup R^{+\smile} &= \{(1,0),(2,0),(0,1),(0,2)\} \\
(R \cup R^{\smile})^{+} &= \{(0,0),(0,1),(0,2),(1,0),(1,1),(1,2),(2,0),(2,1),(2,2)\}
\end{aligned}$$

Here is another question about relations. Suppose you know that R and S are transitive. Does it follow that their composition, the relation you get by first taking an R step and next an S step, is also transitive? The composition of R and S is indicated by $R \circ S$.

Here is a definition of the composition of R and S in predicate logic:

$$\exists z(Rxz \wedge Szy).$$

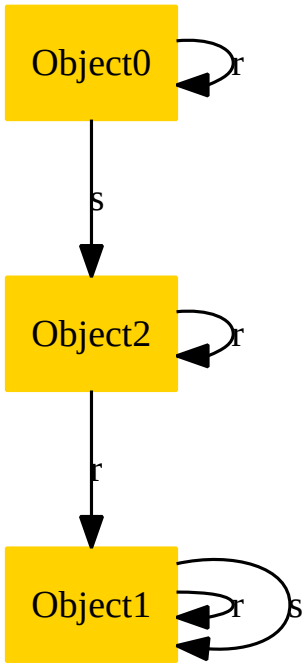
Exercise 10.11 Find a formula of predicate logic stating that if R and S are transitive then their composition is transitive as well.

The answer to exercise 10.11 gives us a rephrasing of our original question: does the formula φ that you constructed have counterexamples (model where it is not true), or not?

The Alloy version of the question is again very succinct. This is because we can state the claim that R is transitive simply as: $R = R^{+}$.

```
sig Object { r,s: set Object }
fact { r = ^r and s = ^s }
assert claim { r.s = ^(r.s) }
check claim
```

Again the system finds counterexamples:



In this example, $R = \{(0,0),(2,2),(2,1),(1,1)\}$ and $S = \{(0,2),(1,1)\}$.

Exercise 10.12 This exercise is about the example relations R and S that were found by Alloy. For these R and S, give $R \circ S$ and give $(R \circ S)^+$. Check that these relations are not the same, so $R \circ S$ is not transitive.

10.5 Conjunctive Normal Form for Predicate Logic

Now suppose we have a predicate logical formula. We will assume that there are no free variables: each variable occurrence is bound by a quantifier. In other words: we assume that the formula is *closed*.

To convert closed formulas of predicate logic to conjunctive normal form, the following steps have to be performed:

(1) Convert to arrow-free form.

(2) Convert to negation normal form by moving $\neg$ signs inwards. This involves the laws of De Morgan, plus the following quantifier principles:

- $\neg\forall x\varphi \leftrightarrow \exists x\neg\varphi$.
- $\neg\exists x\varphi \leftrightarrow \forall x\neg\varphi$.

(3) Standardize variables, in order to make sure that each variable binder $\forall x$ or $\exists x$ occurs only once in the formula. For example, $\forall xPx \vee \exists xQx$ should be changed

to $\forall x Px \vee \exists y Qy$. Or a more complicated example: $\forall x(\exists y(Py \wedge Rxy) \vee \exists y Sxy)$ gets changed to $\forall x(\exists y(Py \wedge Rxy) \vee \exists z Sxz)$. In the standardized version, each variable name x will have exactly one binding quantifier in the formula. This will avoid confusion later, when we are going to drop the quantifiers.

(4) Move all quantifiers to the outside, by using the following equivalences:

- $(\forall x\varphi \wedge \psi) \leftrightarrow \forall x(\varphi \wedge \psi)$,
- $(\forall x\varphi \vee \psi) \leftrightarrow \forall x(\varphi \vee \psi)$.
- $(\exists x\varphi \wedge \psi) \leftrightarrow \exists x(\varphi \wedge \psi)$,
- $(\exists x\varphi \vee \psi) \leftrightarrow \exists x(\varphi \vee \psi)$.

Note that these principles hold because accidental capture of variables is impossible. We standardized the variables, so we may assume that every variable name x has exactly one binding quantifier in the formula. Recall that there are no free variables.

(5) Get rid of existential quantifiers, as follows.

- If the outermost existential quantifier $\exists x$ of the formula is not in the scope of any universal quantifiers, remove it, and replace every occurrence of x in the formula by a fresh constant c.
- If the outermost existential quantifier $\exists x$ of the formula is in the scope of universal quantifiers $\forall y_1$ through $\forall y_n$, remove it, and replace every occurrence of x in the formula by a fresh function $f(y_1, \ldots y_n)$. (Such a function is called a *Skolem function*.)
- Continue like this until there are no existential quantifiers left.

This process is called *skolemization*.

(6) Remove the universal quantifiers.

(7) Distribute disjunction over conjunction, using the equivalences:

- $((\varphi \wedge \psi) \vee \chi) \leftrightarrow ((\varphi \vee \chi) \wedge (\psi \vee \chi))$,
- $(\varphi \vee (\psi \wedge \chi)) \leftrightarrow ((\varphi \vee \chi) \wedge (\varphi \vee \chi))$.

To illustrate the stages of this process, we run through an example. We start with the formula:

$$\forall x(\exists y(Py \vee Rxy) \rightarrow \exists y Sxy).$$

First step: make this arrow-free:

$$\forall x(\neg\exists y(Py \vee Rxy) \vee \exists y Sxy).$$

Second step: move negations inwards:

$$\forall x(\forall y(\neg Py \wedge \neg Rxy) \vee \exists y Sxy).$$

Third step: standardize variables:

$$\forall x(\forall y(\neg Py \wedge \neg Rxy) \vee \exists z Sxz).$$

Fourth step: move quantifiers out:

$$\forall x \forall y \exists z((\neg Py \wedge \neg Rxy) \vee Sxz).$$

Fifth step: skolemization:

$$\forall x \forall y((\neg Py \wedge \neg Rxy) \vee Sxf(x,y)).$$

Sixth step: remove universal quantifiers:

$$((\neg Py \wedge \neg Rxy) \vee Sxf(x,y)).$$

Seventh step, distribute disjunction over conjunction:

$$(\neg Py \vee Sxf(x,y)) \wedge (\neg Rxy \vee Sxf(x,y)).$$

The clause form of the predicate logical formula contains two clauses, and it looks like this:

$$\{\{\neg Py, Sxf(x,y)\}, \{\neg Rxy, Sxf(x,y)\}\}.$$

Exercise 10.13 Stefan

Exercise 10.14 Stefan

10.6 Substitutions

If we want to compute with first order formulas in clause form, it is necessary to be able to handle substitution of terms in such forms. In fact, we will look at the effects of substitutions on terms, on clauses, and on clause forms.

A *variable binding* is a pair consisting of a variable and a term. A binding *binds* the variable to the term. A binding (v,t) is often represented as $v \mapsto t$. A binding is *proper* if it does not bind variable v to term v (the same variable, viewed as a term). A variable substitution is a finite list of proper bindings, satisfying the requirement that no variable v occurs as a lefthanded member in more than one binding $v \mapsto t$.

The substitution that changes nothing is called the *identity substitution*. It is represented by the empty list of variable bindings. We will denote it as ϵ.

The domain of a substitution is the list of all lefthanded sides of its bindings. The range of a substitution is the list of all righthand sides of its bindings. For example, the domain of the substitution $\{x \mapsto f(x), y \mapsto x\}$ is $\{x, y\}$, and its range is $\{x, f(x)\}$.

Substitutions give rise to mappings from terms to terms via the following recursion. Let σ be a substitution. Then a term t either has the form v (the term is a variable) or the form c (the term is a constant) or the form $f(t_1, \ldots, t_n)$ (the term is a function with n argument terms). The result σt of applying the substitution to the term t is given by:

- $\sigma v := \sigma(v)$,
- $\sigma c := c$,
- $\sigma f(t_1, \ldots, t_n) := f(\sigma t_1, \ldots, \sigma t_m)$.

Next, we define the result of applying a substitution σ to a formula φ, again by recursion on the structure of the formula.

- $\sigma P(t_1, \ldots, t_n) := P(\sigma t_1, \ldots, \sigma t_n)$,
- $\sigma(\neg\varphi) := \neg(\sigma\varphi)$,
- $\sigma(\varphi \wedge \psi) := (\sigma\varphi \wedge \sigma\psi)$,
- $\sigma(\varphi \vee \psi) := (\sigma\varphi \vee \sigma\psi)$,
- $\sigma(\varphi \to \psi) := (\sigma\varphi \to \sigma\psi)$,
- $\sigma(\varphi \leftrightarrow \psi) := (\sigma\varphi \leftrightarrow \sigma\psi)$,
- $\sigma(\forall v\varphi) := \forall v\sigma'\varphi$, where σ' is the result of removing the binding for v from σ,
- $\sigma(\exists v\varphi) := \exists v\sigma'\varphi$, where σ' is the result of removing the binding for v from σ.

Exercise 10.15 Stefan

Exercise 10.16 Stefan

The composition of substitution σ with substitution τ should result in the substitution that one gets by applying σ after τ. The following definition has the desired effect.

Definition 10.17 (Composition of substitution representations) Let

$$\theta = [v_1 \mapsto t_1, \ldots, v_n \mapsto t_n] \text{ and } \sigma = [w_1 \mapsto r_1, \ldots, w_m \mapsto r_m]$$

be substitution representations. Then $\theta \cdot \sigma$ is the result of removing from the sequence

$$[w_1 \mapsto \theta(r_1), \ldots, w_m \mapsto \theta(r_m), v_1 \mapsto t_1, \ldots, v_n \mapsto t_n]$$

the bindings $w_1 \mapsto \theta(r_i)$ for which $\theta(r_i) = w_i$, and the bindings $v_j \mapsto t_j$ for which $v_j \in \{w_1, \ldots, w_m\}$.

Exercise 10.18 Prove that this definition gives the correct result.

Applying the recipe for composition to $\{x \mapsto y\} \cdot \{y \mapsto z\}$ gives $\{y \mapsto z, x \mapsto y\}$, applying it to $\{y \mapsto z\} \cdot \{x \mapsto y\}$ gives $\{x \mapsto z, y \mapsto z\}$. This example illustrates the fact that order of application of substitution matters. Substitutions do *not* commute.

Exercise 10.19 Stefan

Exercise 10.20 Stefan

We use the notion of composition to define a relation $\sqsubseteq$ on the set S of all substitutions (for given sets of variables V and terms T), as follows. $\theta \sqsubseteq \sigma$ iff there is a substitution ρ with $\theta = \rho \cdot \sigma$. ($\theta \sqsubseteq \sigma$ is sometimes pronounced as: 'θ is less general than σ.')

The relation $\sqsubseteq$ is **reflexive**. For all θ we have that $\theta = \epsilon \cdot \theta$, and therefore $\theta \sqsubseteq \theta$. The relation is also **transitive**. $\sqsubseteq$ is transitive because if $\theta = \rho \cdot \sigma$ and $\sigma = \tau \cdot \gamma$ then $\theta = \rho \cdot (\tau \cdot \gamma) = (\rho \cdot \tau) \cdot \gamma$, i.e., $\theta \sqsubseteq \gamma$. A relation that is reflexive and transitive is called a **pre-order**, so what we have just shown is that $\sqsubseteq$ is a pre-order.

10.7 Unification

If we have two expressions A and B (where A, B can be terms, or formulas, or clauses, or clause forms), each containing variables, then we are interested in the following questions:

- Is there a substitution θ that makes A and B equal?
- How do we find such a substitution in an efficient way?

We introduce some terminology for this. The substitution θ *unifies* expressions A and B if $\theta A = \theta B$. The substitution θ *unifies* two sequences of expressions $(A_1, \ldots, A_n)$ and $(B_1, \ldots, B_n)$ if, for $1 \leq i \leq n$, θ unifies A_i and B_i. Note that unification of pairs of atomic formulas reduces to unification of sequences of terms, for two atoms that start with a different predicate symbol do not unify, and two atoms $P(t_1, \ldots, t_n)$ and $P(s_1, \ldots, s_n)$ unify iff the sequences $(t_1, \ldots, t_n)$ and $(s_1, \ldots, s_n)$ unify.

What we are going to need to apply resolution reasoning (Section 10.3) to predicate logic is unification of pairs of atomic formulas.

For example, we want to find a substitution that unifies the pair

$$P(x, g(a, z)), P(g(y, z), x).$$

In this example case, such unifying substitutions exist. A possible solution is

$$\{x \mapsto g(a, z), y \mapsto a\}.$$

for applying this substitution gives $P(g(a,z),g(a,z))$. Another solution is

$$\{x \mapsto g(a,b), y \mapsto a, z \mapsto b\}.$$

In this case, the second solution is an instance of the first, for

$$\{x \mapsto g(a,b), y \mapsto a, z \mapsto b\} \sqsubseteq \{x \mapsto g(a,z), y \mapsto a\},$$

because

$$\{x \mapsto g(a,b), y \mapsto a, z \mapsto b\} = \{z \mapsto b\} \cdot \{x \mapsto g(a,z), y \mapsto a\}.$$

So we see that solution $\{x \mapsto g(a,z), y \mapsto a\}$ is more general than solution $\{x \mapsto g(a,b), y \mapsto a, z \mapsto b\}$.

If a pair of atoms is unifiable, it is useful to try and identify a solution that is as general as possible, for the more general a solution is, the less unnecessary bindings it contains. These considerations motivate the following definition.

Definition 10.21 If θ is a unifier for a pair of expressions (a pair of sequences of expressions), then θ is called an mgu (a most general unifier) if $\sigma \sqsubseteq \theta$ for every unifier σ for the pair of expressions (the pair of sequences of expressions).

In the above example, $\{x \to g(a,z), y \mapsto a\}$ is an mgu for the pair

$$P(x,g(a,z)), P(g(y,z),x).$$

The **Unification Theorem** says that if a unifier for a pair of sequences of terms exists, then an mgu for that pair exists as well. Moreover, there is an algorithm that produces an mgu for any pair of sequences of terms in case these sequences are unifiable, and otherwise ends with failure.

We will describe the *unification algorithm* and prove that it does what it is supposed to do. This constitutes the proof of the theorem.

We give the algorithm in stages.

First we define unification of terms *UnifyTs*, in three cases.

- Unification of two variables x and y gives the empty substitution if the variables are identical, and otherwise a substitution that binds one variable to the other.
- Unification of x to a non-variable term t fails if x occurs in t, otherwise it yields the binding $\{x \mapsto t\}$.
- Unification of $f\bar{t}$ and $g\bar{r}$ fails if the two variable names are different, otherwise it yields the return of the attempt to do term list unification on $\bar{t}$ and $\bar{r}$.

If unification succeeds, a unit list containing a representation of a most general unifying substitution is returned. Return of the empty list indicates unification failure.

Unification of term lists (*UnifyTlists*):

- Unification of two empty term lists gives the identity substitution.
- Unification of two term lists of different length fails.
- Unification of two term lists $t_1, \ldots, t_n$ and $r_1, \ldots, r_n$ is the result of trying to compute a substitution $\sigma = \sigma_n \circ \cdots \circ \sigma_1$, where
  - σ_1 is a most general unifier of t_1 and r_1,
  - σ_2 is a most general unifier of $\sigma_1(t_2)$ and $\sigma_1(r_2)$,
  - σ_3 is a most general unifier of $\sigma_2\sigma_1(t_3)$ and $\sigma_2\sigma_1(r_3)$,
  - and so on.

Our task is to show that these two unification functions do what they are supposed to do: produce a unit list containing an mgu if such an mgu exists, produce the empty list in case unification fails.

The proof consists of a Lemma and two Theorems. The Lemma is needed in Theorem 10.23. The Lemma establishes a simple property of mgu's. Theorem 10.24 establishes the result.

Lemma 10.22 If σ_1 is an mgu of t_1 and s_1, and σ_2 is an mgu of

$$(\sigma_1 t_2, \ldots, \sigma_1 t_n) \text{ and } (\sigma_1 s_2, \ldots, \sigma_1 s_n),$$

then $\sigma_2 \cdot \sigma_1$ is an mgu of $(t_1, \ldots, t_n)$ and $(s_1, \ldots, s_n)$.

Proof. Let θ be a unifier of $(t_1, \ldots, t_n)$ and $(s_1, \ldots, s_n)$. Given this assumption, we have to show that $\sigma_2 \cdot \sigma_1$ is more general than θ.

By assumption about θ we have that $\theta t_1 = \theta s_1$. Since σ_1 is an mgu of t_1 and s_1, there is a substitution ρ with $\theta = \rho \cdot \sigma_1$.

Again by assumption about θ, it holds for all i with $1 < i \leq n$ that $\theta t_i = \theta s_i$. Since $\theta = \rho \cdot \sigma_1$, it follows that

$$(\rho \cdot \sigma_1)t_i = (\rho \cdot \sigma_1)s_i,$$

and therefore,

$$\rho(\sigma_1 t_i) = \rho(\sigma_1 s_i).$$

Since σ_2 is an mgu of $(\sigma_1 t_2, \ldots, \sigma_1 t_n)$ and $(\sigma_1 s_2, \ldots, \sigma_1 s_n)$, there is a substitution ν with $\rho = \nu \cdot \sigma_2$. Therefore,

$$\theta = \rho \cdot \sigma_1 = (\nu \cdot \sigma_2) \cdot \sigma_1 = \nu \cdot (\sigma_2 \cdot \sigma_1).$$

This shows that $\sigma_2 \cdot \sigma_1$ is more general than θ, which establishes the Lemma. □

Theorem 10.23 shows, by induction on the length of term lists, that if *unifyTs* t s does what it is supposed to do, then *unifyTlists* also does what it is supposed to do.

Theorem 10.23 Suppose *unifyTs t s* yields a unit list containing an mgu of t and s if the terms are unifiable, and otherwise yields the empty list. Then *unifyTlists* $\bar{t}$ $\bar{s}$ yields a unit list containing an mgu of $\bar{t}$ and $\bar{s}$ if the lists of terms $\bar{t}$ and $\bar{s}$ are unifiable, and otherwise produces the empty list.

Proof. If the two lists have different lengths then unification fails.

Assume, therefore, that $\bar{t}$ and $\bar{s}$ have the same length n. We proceed by induction on n.

Basis $n = 0$, i.e., both $\bar{t}$ and $\bar{s}$ are equal to the empty list. In this case the ϵ substitution unifies $\bar{t}$ and $\bar{s}$, and this is certainly an mgu.

Induction step $n > 0$. Assume $\bar{t} = (t_1, \ldots, t_n)$ and $\bar{s} = (s_1, \ldots, s_n)$, with $n > 0$. Then $\bar{t} = t_1 : (t_2, \ldots, t_n)$ and $\bar{s} = s_1 : (s_2, \ldots, s_n)$, where : expresses the operation of putting an element in front of a list.

What the algorithm does is:

(1) It checks if t_1 and s_1 are unifiable by calling *unifyTs* t_1 s_1. By the assumption of the theorem, *unifyTs* t_1 s_1. yields a unit list (σ_1), with σ_1 an mgu of t_1 and s_1 if t_1 and s_1 are unifiable, and yields the empty list otherwise. In the second case, we know that the lists $\bar{t}$ and $\bar{s}$ are not unifiable, and indeed, in this case *unifyTlists* will produce the empty list.

(2) If t_1 and s_1 have an mgu σ_1, then the algorithm tries to unify the lists

$$(\sigma_1 t_2, \ldots, \sigma_1 t_n) \text{ and } (\sigma_1 s_2, \ldots, \sigma_1 s_n),$$

i.e., the lists of terms resulting from applying σ_1 to each of $(t_2, \ldots, t_n)$ and each of $(s_2, \ldots, s_n)$. By induction hypothesis we may assume that applying `unifyTlists` to these two lists produces a unit list (σ_2), with σ_2 an mgu of the lists, if the two lists are unifiable, and the empty list otherwise.

(3) If σ_2 is an mgu of the two lists, then the algorithm returns a unit list containing $\sigma 2 \cdot \sigma_1$. By Lemma 10.22, $\sigma 2 \cdot \sigma_1$ is an mgu of $\bar{t}$ and $\bar{s}$.

□

Theorem 10.24 clinches the argument. It proceeds by structural induction on terms. The induction hypothesis will allow us to use Theorem 10.23.

Theorem 10.24 The function *unifyTs t s* either yields a unit list (γ) or the empty list. In the former case, γ is an mgu of t and s. In the latter case, t and s are not unifiable.

Proof. Structural induction on the complexity of (t, s). There are 4 cases.

1. Both terms are variables, i.e., t equals x, s equals y. In this case, if x and y are identical, the ϵ substitution is surely an mgu of t and s. This is what the algorithm yields. If x and y are different variables, then the substitution $\{x \mapsto y\}$ is an mgu of x and y. For suppose $\sigma x = \sigma y$. Then $\sigma x = (\sigma \cdot \{x \mapsto y\})x$, and for all z different from x we have $\sigma z = (\sigma \cdot \{x \mapsto y\})z$. So $\sigma = \sigma \cdot \{x \mapsto y\}$.

2. $t = x$ and s is not a variable. If x is not an element of the variables of s, then $\{x \mapsto s\}$ is an mgu of t and s. For if $\sigma x = \sigma s$, then $\sigma x = (\sigma \cdot \{x \mapsto s\})x$, and for all variables z different from x we have that $\sigma z = (\sigma \cdot \{x \mapsto s\})z$. $\sigma = \sigma \cdot \{x \mapsto s\}$. If x is an element of the variables of s, then unification fails (and this is what the algorithm yields).

3. $s = x$ and t not a variable. Similar to case 2.

4. $t = f(\bar{t})$ and $s = g(\bar{s})$. Then t and s are unifiable iff (i) f equals g and (ii) the term lists $\bar{t}$ and $\bar{s}$ are unifiable. Moreover, ν is an mgu of t and s iff f equals g and ν is an mgu of $\bar{t}$ and $\bar{s}$.

By the induction hypothesis, we may assume for all subterms t' of t and all subterms s' of s that *unifyTs t' s'* yields the empty list if t' and s' do not unify, and a unit list (ν), with ν an mgu of t' and s' otherwise. This means the condition of Theorem 10.23 is fulfilled, and it follows that *unifyTlists* $\bar{t}$ $\bar{s}$ yields (ν), with ν an mgu of $\bar{t}$ and $\bar{s}$, if the term lists $\bar{t}$ and $\bar{s}$ unify, and *unifyTlists* $\bar{t}$ $\bar{s}$ yields the empty list if the term lists do not unify.

This establishes the Theorem. □

Some examples of unification attempts:

- unifyTs x $(f(x)$ yields $()$.

- unifyTs x $(f(y)$ yields $(\{x \mapsto y\})$.
- unifyTs $g(x, a)$ $g(y, x)$ yields $(\{x \mapsto a, y \mapsto a\})$.

Further examples are in the exercises.

Exercise 10.25 Stefan

Exercise 10.26 Stefan

Exercise 10.27 Stefan

10.8 Resolution with Unification

Suppose we have clausal forms for predicate logic. Then we can adapt the resolution rule to predicate logic by combining resolution with unification, as follows. Assume that $C_1 \cup \{P\bar{t}\}$ and $C_2 \cup \{\neg P\bar{s}\}$ are predicate logical clauses. The two literals $P\bar{t}$ and $P\bar{s}$ need not be the same in order to apply resolution to the clauses. It is enough that $P\bar{t}$ and $P\bar{s}$ are *unifiable*.

For what follows, let us assume that the clauses in a predicate logical clause form do not have variables in common. This assumption is harmless: see Exercise 10.28.

Exercise 10.28 Suppose C and C' are predicate logical clauses, and they have a variable x in common. Show that it does not affect the meaning of the clause form $\{C, C'\}$ if we replace the occurrence(s) of x in C' by occurrences of a fresh variable z ("freshness" of z means that z occurs in neither C nor C'.)

Assume that $C_1 \cup \{P\bar{t}\}$ and $C_2 \cup \{\neg P\bar{s}\}$ do not have variables in common. Then the following inference rule is sound:

Resolution Rule with Unification

$$\frac{C_1 \cup \{P\bar{t}\} \qquad \{\neg P\bar{s}\} \cup C_2}{\theta C_1 \cup \theta C_2} \ \theta \text{ is mgu of } \bar{t} \text{ and } \bar{s}$$

Here is an example application:

$$\frac{\{Pf(y), Qg(y)\} \quad \{\neg Pf(g(a)), Rby\}}{\{Qg(g(a)), Rbg(a)\}} \ \text{mgu } \{y \mapsto g(a)\} \text{ applied to } Pf(y) \text{ and } Pf(g(a))$$

It is also possible to use unification to 'simplify' individual clauses. If $P\bar{t}$ and $P\bar{s}$ (or $\neg P\bar{t}$ and $\neg P\bar{s}$) occur in the same clause C, and θ is an mgu of $\bar{t}$ and $\bar{s}$, then θC is called a **factor** of C. The following inference rules identify literals by means of factorisation:

Factorisation Rule (pos)

$$\frac{C_1 \cup \{P\bar{t}, P\bar{s}\}}{\theta(C_1 \cup \{P\bar{t}\})} \ \theta \text{ is mgu of } \bar{t} \text{ and } \bar{s}$$

Factorisation Rule (neg)

$$\frac{C_1 \cup \{\neg P\bar{t}, \neg P\bar{s}\}}{\theta(C_1 \cup \{\neg P\bar{t}\})} \ \theta \text{ is mgu of } \bar{t} \text{ and } \bar{s}$$

An example application:

$$\frac{\{Px, Pf(y), Qg(y)\}}{\{Pf(y), Qg(y)\}} \text{ mgu } \{x \mapsto f(y)\} \text{ applied to } Px \text{ and } Pf(y))a$$

Resolution and factorisation can also be combined, as in the following example:

$$\frac{\dfrac{\{Px, Pf(y), Qg(y)\}}{\{Pf(y), Qg(y)\}} \text{ factorisation} \qquad \{\neg Pf(g(a)), Rby\}}{\{Qg(g(a)), Rbg(a)\}} \text{ resolution}$$

Computation with first order logic uses these rules, together with a **search strategy** for selecting the clauses and literals to which resolution and unification are going to be applied. A particularly simple strategy is possible if we restrict the format of the clauses in the clause forms.

It can be proved (although we will not do so here) that resolution and factorisation for predicate logic form a *complete* calculus for predicate logic. What this means is that a clause form F is unsatisfiable if and only if there exists a deduction of the empty clause $[]$ from F by means of resolution and factorisation.

On the other hand, there is an important difference with the case of propositional logic. Resolution refutation is a *decision method* for (un)satisfiability in propositional logic. In the case of predicate logic, this cannot be the case, for predicate logic has no decision mechanism. Resolution/factorisation refutation does not decide predicate logic. More precisely, if a predicate logical clause F is unsatisfiable, then there exists a resolution/factorisation derivation of $[]$ from F, but if F is satisfiable, then the derivation process may never stop, as the possibilities of finding ever new instantiations by means of unification are inexhaustible.

10.9 Prolog

Prolog, which derives its name from *programming with logic*, is a general purpose programming language that is popular in artificial intelligence and computational linguistics, and that derives its force from a clever search strategy for a particular kind of restricted clause form for predicate logic.

Alain Colmerauer

The language was conceived in the 1970s by a group around Alain Colmerauer in Marseille. The first Prolog system was developed in 1972 by Alain Colmerauer and Phillipe Roussel. A well known public domain version of Prolog us SWI-Prolog, developed in Amsterdam by Jan Wielemaker. See `http://www.swi-prolog.org/`.

Jan Wielemaker

Definition 10.29 A clause with just one positive literal is called a *program clause*. A clause with only negative literals is called a *goal clause*.

A program clause $\{\neg A_1, \ldots, \neg A_n, B\}$ can be viewed as an implication $(A_1 \wedge \cdots, A_n) \rightarrow B$. A goal clause $\{\neg A_1, \ldots, \neg A_n\}$ can be viewed as a degenerate implication $(A_1 \wedge \cdots, A_n) \rightarrow []$, where $[]$ is the empty clause (expressing a contradiction). Goal and program clauses together constitute what is is called *pure Prolog*. The computation strategy of Prolog consists of combining a goal clause with a number of program clauses in an attempt to derive the empty clause. Look at the goal clause like this:

$$(A_1 \wedge \cdots, A_n) \rightarrow [].$$

From this, [] can be derived if we manage to derive each of $A_1, \ldots, A_n$ from the Prolog program clauses. An example will clarify this. In the following example of a pure Prolog program, we use the actual Prolog notation, where predicates are lower case, variables are upper case, and implications $(A_1 \wedge \cdots, A_n) \rightarrow B$ are written backwards, as $B :- A_1, \ldots, A_n$.

```
plays(heleen,X) :- haskeys(X).
plays(heleen,violin).
plays(hans,cello).
plays(jan,clarinet).

haskeys(piano).
haskeys(accordeon).
haskeys(keyboard).
haskeys(organ).

woodwind(clarinet).
woodwind(recorder).
woodwind(oboe).
woodwind(bassoon).
```

Each line is a program clause. All clauses except one consist of a single positive literal. The exception is the clause `plays(heleen,X) :- haskeys(X).` This is the Prolog version of $\forall x(H(x) \rightarrow P(h, x))$. Here is an example of interaction with this database (read from a file `music.pl`) in SWI-Prolog:

```
[jve@pidgeot lia]$ pl
Welcome to SWI-Prolog (Multi-threaded, 64 bits, Version 5.6.64)
Copyright (c) 1990-2008 University of Amsterdam.
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

?- [music].
% music compiled 0.00 sec, 3,328 bytes
true.

?- plays(heleen,X).
```

The last line constitutes the Prolog query. The system now computes a number of answers, and we can use ; after each answer to prompt for more, until the list of answers is exhausted. This is what we get:

```
X = piano ;
X = accordeon ;
X = keyboard ;
X = organ ;
X = violin.
```

Prolog queries can also be composite:

```
?- woodwind(X),plays(Y,X).
X = clarinet,
Y = jan ;
false.

?-
```

The strategy that Prolog uses to compute answers is resolution refutation. Take the first query as an example. The Prolog system combines the database clauses (the program clauses in the file `music.pl`) with the goal clause `plays(heleen,X)` $\rightarrow$ [], and sure enough, the system can derive the empty clause [] from this, in quite a number of ways. Each derivation involves a unifying substitution, and these substitutions are what the system computes for us. The exercises to follow invite you to play a bit more with Prolog programming.

Exercise 10.30 Stefan

Exercise 10.31 Stefan

Exercise 10.32 Stefan

Exercise 10.33 Stefan

Exercise 10.34 Stefan

Summary *After having finished this chapter you can check whether you have mastered the material by answering the following questions:*

- *What is the definition of clausal form for propositional logic?*
- *How can formulas of propositional logic be translated into clausal form?*
- *How does the resolution rule work for propositional logic, and why is it sound?*
- *What are SAT solvers? How do they work?*
- *What is the definition of clausal form for predicate logic?*

- *How can formulas of predicate logic be translated into clausal form?*
- *How can variable substitutions be represented as finite sets of bindings?*
- *How are substitutions composed?*
- *What does it mean that one substitution is more general than another one?*
- *What is an mgu?*
- *What is unification? What does the unification algorithm do?*
- *What is the rule of resolution with unification? Why is it sound?*
- *What is the rule of factorisation? Why is it sound?*
- *What are program clauses and goal clauses?*
- *What is the computation mechanism behind Prolog?*

Appendices

Appendix A

Sets, Relations and Functions

Summary

This chapter explains the basics of formal set notation, and gives an introduction to relations and functions. The chapter ends with a short account of the principle of proof by mathematical induction.

A.1 Sets and Set Notation

Many mathematical notions — some say all mathematical notions — can be defined in terms of the fundamental concept of a set. This is good reason for starting with some basic set theory.

A set is a collection of definite, distinct objects. Examples are the set of colours of the Dutch flag, or the set of letters of the Greek alphabet. Yet another example is the set of even natural numbers greater than seven. And so on.

The elements of a set are also called its *members*. To indicate that a is an element of a set A we write $a \in A$. To deny that a is an element of a set A we write $a \notin A$. The symbol $\in$ is the symbol for membership.

The elements of a set can be anything: words, colours, people, numbers. The elements of a set can also themselves be sets. The set consisting of the set of even natural numbers and the set of odd natural numbers is an example. This set has two elements; each of these elements has itself an infinite number of elements.

To check whether two sets are the same one has to check that they have the same elements. The fact that membership is all there is to set identity, or that sets are fully determined by their members, is called the *principle of extensionality*. It follows that to check that two sets A and B are identical, one has to check two things:

- does it hold that every element a of A is also an element of B, and
- does it hold that every element b of B is also an element of A?

To specify a set, there are several methods: give a list of its members, as in 'the set having the numbers 1, 2 and 3 as its only members', give some kind of semantic description, as in 'the set of colours of the Dutch flag', or separate out a set from a larger set by means of a suitable restriction. This last method is called the method of *set comprehension*. Here is an example: the odd natural numbers are the natural numbers with the property that division by 2 leaves a remainder of 1. We can express this by means of the pattern $2n+1$, as follows:

$$O = \{2n + 1 \mid n \in \mathbb{N}\}.$$

The braces are also used to list the members of a finite set:

$$D = \{\text{red}, \text{white}, \text{blue}\}.$$

Mentioning a set element more than once does not make a difference. The set

$$\{\text{white}, \text{blue}, \text{white}, \text{red}\}$$

is identical to the set D, for it has the same members.

Another way of specifying sets is by means of operations on other sets. An example is the following definition of the odd natural numbers:

$$E = \mathbb{N} - O.$$

Here $\mathbb{N}-O$ is the set of all elements of $\mathbb{N}$ that are not members of O. Equivalent definitions are the following:

$$E = \{n \in \mathbb{N} \mid n \notin O\}$$

or

$$E = \{2n \mid n \in \mathbb{N}\}.$$

Some important sets have special names. $\mathbb{N}$ is an example. Another example is $\mathbb{Z}$, for the set of integer numbers. Yet another example is the set without any members. Because of the principle of extensionality there can be only one such set. It is called $\emptyset$ or the empty set.

If every member of a set A is also a member of set B we say that A is a subset of B, written as $A \subseteq B$. If $A \subseteq B$ and $B \subseteq A$ then it follows by the principle of extensionality that A and B are the same set. Conversely, if $A = B$ then it follows by definition that $A \subseteq B$ and $B \subseteq A$.

Exercise A.1 Explain why $\emptyset \subseteq A$ holds for every set A.

Exercise A.2 Explain the difference between $\emptyset$ and $\{\emptyset\}$.

The *complement* of a set A, with respect to some fixed universe, or: domain, U with $A \subseteq U$, is the set consisting of all objects in U that are not elements of A. The complement set is written as $\overline{A}$. It is defined as the set $\{x \mid x \in U, x \notin A\}$. For example, if we take U to be the set $\mathbb{N}$ of natural numbers, then the set of even numbers is the complement of the set of odd numbers and vice versa.

A.2 Relations

By a relation we mean a meaningful link between people, things, objects, whatever. Usually, it is quite important what kind of relationship we have in mind.

Formally, we can describe a *relation* between two sets A and B as a collection of ordered pairs (a, b) such that $a \in A$ and $b \in B$. An ordered pair is, as the name already gives away, a collection of two distinguishable objects, in which the order plays a role. E.g., we use $(\textit{Bill}, \textit{Hillary})$ to indicate the ordered pair that has *Bill* as its first element and *Hillary* as its second element. This is different from the pair $(\textit{Hillary}, \textit{Bill})$ where Bill plays second fiddle.

The notation for the set of all ordered pairs with their first element taken from A and their second element taken from B is $A \times B$. This is called the *Cartesian product* of A and B. A relation between A and B is a subset of $A \times B$.

The Cartesian product of the sets $A = \{a, b, \ldots, h\}$ and $B = \{1, 2, \ldots, 8\}$, for example, is the set

$$A \times B = \{(a,1), (a,2), \ldots, (b,1), (b,2), \ldots, (h,1), (h,2), \ldots, (h,8)\}.$$

This is the set of positions on a chess board. And if we multiply the set of chess colours $C = \{\text{White}, \text{Black}\}$ with the set of chess figures,

$$F = \{\text{King}, \text{Queen}, \text{Knight}, \text{Rook}, \text{Bishop}, \text{Pawn}\},$$

we get the set of chess pieces $C \times F$. If we multiply this set with the set of chess positions, we get the set of piece positions on the board, with $(\text{White}, \text{King}, (e,1))$ indicating that the white king occupies square $e1$. To get the set of moves on a chess board, take $((C \times F) \times ((A \times B) \times (A \times B)))$, and read $((\text{White}, \text{King}, ((e,1), (f,2))$ as 'white king moves from $e1$ to $f2$', but bear in mind that not all moves in $((C \times F) \times ((A \times B) \times (A \times B)))$ are legal in the game.

$A \times A$ is sometimes also denoted by A^2. Similarly for $A \times A \times A$ and A^3, and so on.

As an example of a relation as a set of ordered pairs consider the relation of authorship between a set A of authors and a set B of books. This relation associates with every author the book(s) he or she wrote.

Sets of ordered pairs are called binary relations. We can easily generalize this to sets of triples, to get so-called ternary relations, to sets of quadruples, and so on. An example

of a ternary relation is that of borrowing something from someone. This relation consists of triples, or: 3-tuples, (a, b, c), where a is the borrower, b is the owner, and c is the thing borrowed. In general, an n-ary relation is a set of n-tuples (ordered sequences of n objects). We use A^n for the set of all n-tuples with all elements taken from A.

Unary relations are called *properties*. A property can be represented as a set, namely the set that contains all entities having the property. For example, the property of being divisible by 3, considered as a property of integer numbers, corresponds to the set $\{\ldots, -9, -6, -3, 0, 3, 6, 9, \ldots\}$.

An important operation on binary relations is composition. If R and S are binary relations on a set U, i.e. $R \subseteq U^2$ and $S \subseteq U^2$, then the composition of R and S, notation $R \circ S$, is the set of pairs (x, y) such that there is some z with $(x, z) \in R$ and $(z, y) \in S$. E.g., the composition of $\{(1, 2), (2, 3)\}$ and $\{(2, 4), (2, 5)\}$ is $\{(1, 4), (1, 5)\}$.

Exercise A.3 What is the composition of $\{(n, n + 2) \mid n \in \mathbb{N}\}$ with itself?

Another operation on binary relations is converse. If R is a binary relation, then its converse (or: inverse) is the relation given by $R^{\smile} = \{(y, x) \mid (x, y) \in R\}$. The converse of the relation 'greater than' on the natural numbers is the relation 'smaller than' on the natural numbers. If a binary relation has the property that $R^{\smile} \subseteq R$ then R is called *symmetric*. It is also denoted as

$$\forall x \forall y (Rxy \rightarrow Ryx). \tag{A.1}$$

Exercise A.4 Show that it follows from $R^{\smile} \subseteq R$ that $R = R^{\smile}$.

If U is a set, then the relation $I = \{(x, x) \mid x \in U\}$ is called the identity relation on U. If a relation R on U has the property that $I \subseteq R$, i.e. if every element of U stands in relation R to itself, then R is called *reflexive*. The relation $\leq$ ('less than or equal') on the natural numbers is reflexive, the relation $<$ ('less than') is not. The relation $A^2 - I$ is the set of all pairs $(x, y) \in A^2$ with $x \neq y$. If A is the set $\{a, bc\}$, then $A^2 - I$ gives the following relation:

$$\{(a, b), (a, c), (b, a), (b, c), (c, a)(c, b)\}.$$

A relation R is called *transitive* if it holds for all x, y, z that if $(x, y) \in R$ and $(y, z) \in R$, then also $(x, z) \in R$. To say that the relation of friendship is transitive boils down to saying that it holds for anyone that the friends of their friends are their friends.

Exercise A.5 Which of the following relations are transitive?

(1) $\{(1, 2), (2, 3), (3, 4)\}$

(2) $\{(1, 2), (2, 3), (3, 4), (1, 3), (2, 4)\}$

(3) $\{(1, 2), (2, 3), (3, 4), (1, 3), (2, 4), (1, 4)\}$

(4) $\{(1, 2), (2, 1)\}$

(5) $\{(1,1),(2,2)\}$

The next exercise shows that transitivity can be expressed in terms of relational composition.

Exercise A.6 Check that a relation R is transitive if and only if it holds that $R \circ R \subseteq R$.

Exercise A.7 Can you give an example of a transitive relation R for which $R \circ R = R$ does not hold?

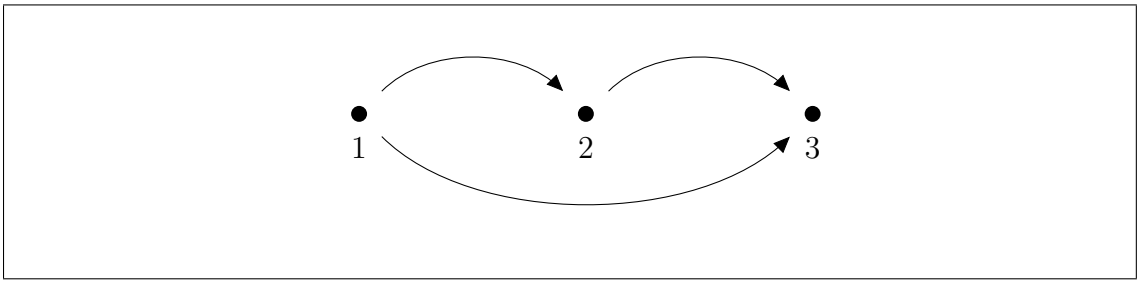
A.3 Back and Forth Between Sets and Pictures

A domain of discourse with a number of 1-place and 2-place predicates on is in fact a set of entities with certain designated subsets (the 1-place predicates) and designated sets of pairs of entities (the 2-place predicates).

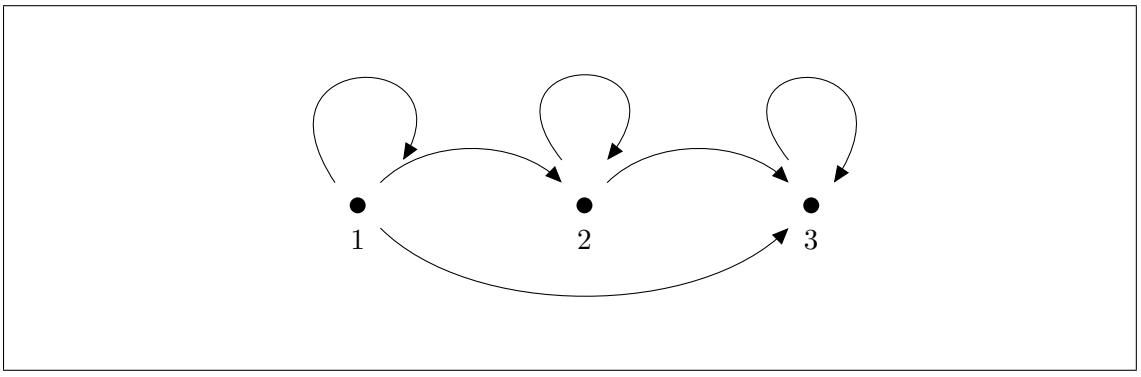
Relations are sets of pairs, and it is useful to acquire the skill to mentally go back and forth between sets-of-pairs representation and picture representation. Take the following simple example of a relation on the set $\{1,2,3\}$.

$$\{(1,2),(1,3),(2,3)\}. \tag{A.2}$$

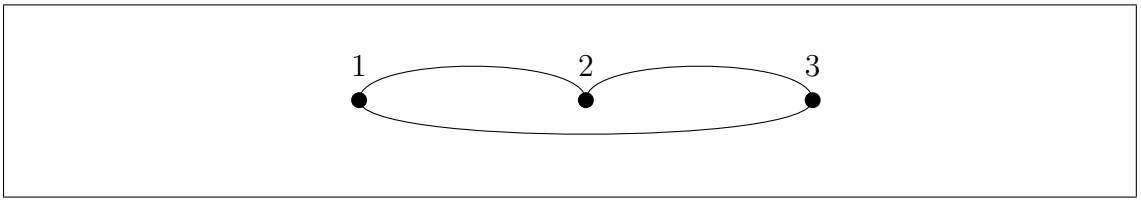
Here is the corresponding picture:



Exercise A.8 Give the set of pairs that constitutes the relation of the following picture:



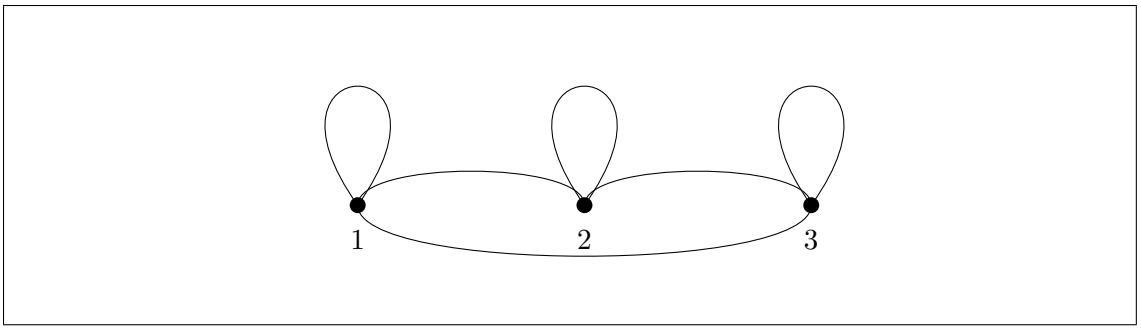
For another example, consider the picture:



No arrowheads are drawn, which indicates that the pictured relation is symmetric. Here is the representation of the same relation as a set of pairs:

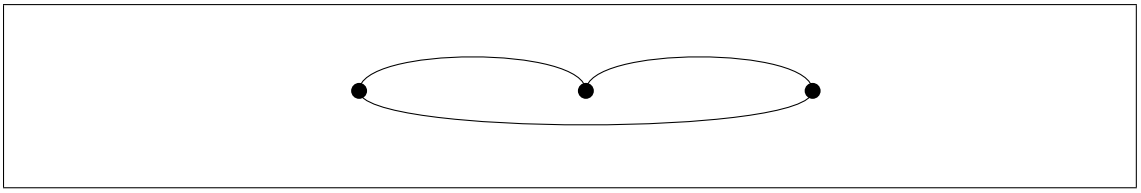
$$\{(1,2),(2,1),(1,3),(3,1),(2,3),(3,2)\}.$$

Exercise A.9 Give the representation of the pictured relation as a set of pairs:



A.4 Relational Properties

Talking about pictures with predicate logic is very useful to develop a clear view of what relational properties the predicate logical formulas express. Predicate logic is very precise, and it takes practice to get used to this precision. Consider the following picture.

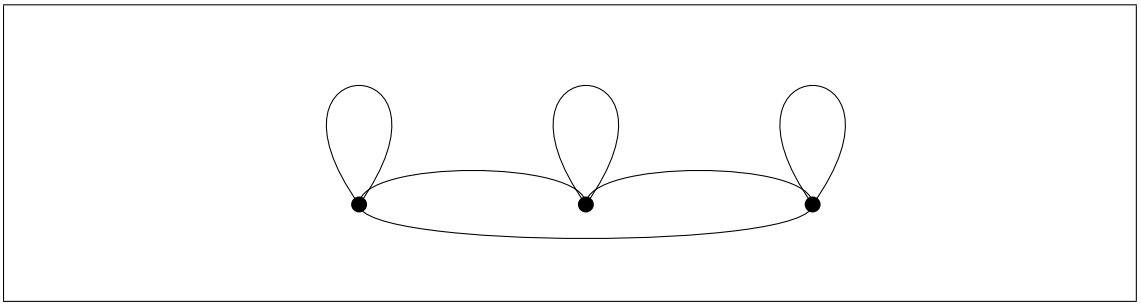


A relation is *transitive* if from the facts that there are links from x to y and a link from y to z it follows that there is a link from x to y. Here is a formula for this:

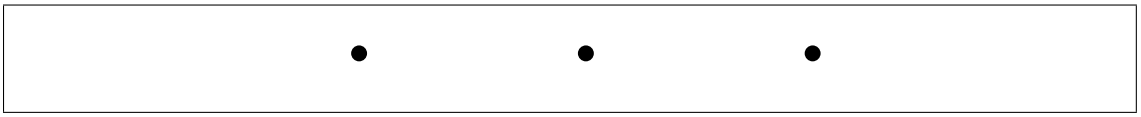
$$\forall x \forall y \forall z((Rxy \wedge Ryz) \rightarrow Rxz). \tag{A.3}$$

Let us check whether the link relation in the last picture is transitive. It may seem at first sight that it is, for what transitivity expresses is that if you can go from x to z by first taking an R step from x to y and next another R step from y to z, between, then there is also a direct R step from x to y. This seems indeed to be the case in the picture. But there is a snag. In reasoning like this, we assume that the three points x, y and z are all *different*. But this is not what the formula says. Take any two different points in the picture. Surely there is a link from the first point to the second. But the linking relation is symmetric: it goes in both directions. Therefore there also is a link from the second point back to the first. But this means that the first point has to be R related to itself, and it isn't. So the relation in the picture is not transitive after all.

Can we also come up with a picture of three points with a symmetric linking relation, where the relation is transitive? Yes, there are several possibilities. Here is the first one:



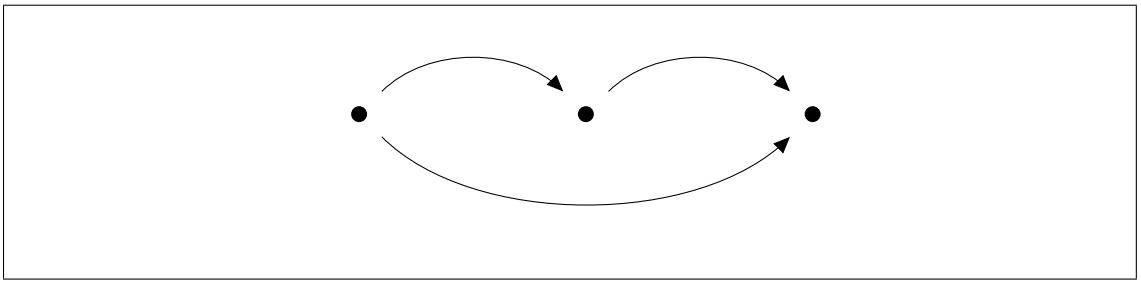
But there is another possibility. Take the following picture:



This is a picture where the link relation is empty. There are no links, so it trivially holds that if one can get from a point to a point via two links, then one can also get there with a single link. So the empty relation is transitive.

Exercise A.10 Give all the transitive link relations on a domain consisting of three individuals, on the assumption that the link relation is symmetric. We have already seen two examples: the empty relation (no points are linked) and the total relation (all points are linked). What are the other possibilities? Draw pictures!

The relations in the pictures above were all symmetric: links were the same in both directions. The following picture with arrows gives a relation that is not symmetric. We need the arrows, for now the directions of the links matter:



Again we use R to refer to the binary relation in the picture. Again we can ask if the relation of the picture is transitive. This time the answer is 'yes'. If we can get from x to y with two $\longrightarrow$ steps, then we can also get from x to y with a single step.

Not only is the relation in the picture not symmetric, but something stronger holds:

$$\forall x \forall y (Rxy \rightarrow \neg Ryx). \tag{A.4}$$

Formula (A.4) expresses that the relation R is *asymmetric*.

Exercise A.11 Give an example of a binary relation on a domain of three objects that it is neither symmetric nor asymmetric.

The relation in the current picture also has another property, called *irreflexivity*:

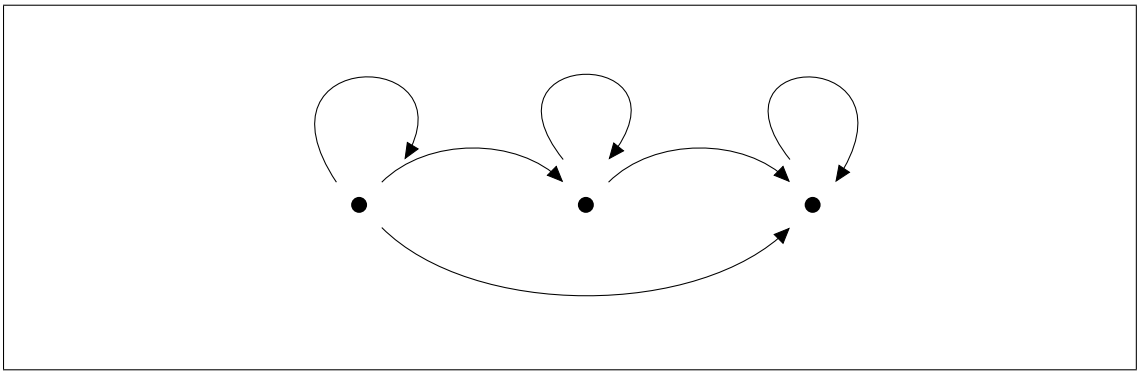
$$\forall x \neg Rxx. \tag{A.5}$$

This expresses that the R relation does not have self loops. We say: the relation is *irreflexive*

The dual to *irreflexivity* is the property of having *all* self loops. This is called *reflexivity*:

$$\forall x Rxx. \tag{A.6}$$

Here is an example of a reflexive relation:



Exercise A.12 Show that any asymmetric relation has to be irreflexive. (Hint: assume that a relation is asymmetric, and suppose it contains a loop (x, x). Why is this impossible?)

A binary relation R is called an *equivalence relation* if R has the following three properties: (i) R is reflexive, (ii) R is symmetric, (iii) R is transitive.

Exercise A.13 Give all equivalence relations on a domain consisting of three objects. Draw pictures!

Exercise A.14 Consider the three predicate logical sentences (A.1), (A.3) and (A.6), These sentences together express that a certain binary relation R is an equivalence relation: symmetric, transitive and reflexive. Show that none of these sentences is semantically entailed by the other ones by choosing for each pair of sentences a model (situation) that makes these two sentences true but makes the third sentence false. In other words: find three examples of binary relations, each satisfying just two of the properties in the list (A.1), (A.3) and (A.6). This shows, essentially, that the definition of being an equivalence cannot be simplified (why?).

Exercise A.15 Consider the following predicate logical formulas:

- $\forall x \forall y (Rxy \rightarrow \neg Ryx)$ (R is asymmetric)
- $\forall x \exists y Rxy$ (R is serial)
- $\forall x \forall y \forall z ((Rxy \wedge Ryz) \rightarrow Rxz)$ (R is transitive).

Take any situation with a non-empty domain of discourse, with a binary relation on it. Show: if the three formulas are true of this situation, then the domain of discourse must be *infinite*. (Hint: start with a domain consisting of a single individual d_1. Then by seriality there has to be an R-successor to d_1. Suppose we take d_1 as its own R-successor. Then this would get us in conflict with we are in conflict with asymmetry, by Exercise **??**. So there has to be a d_2 with (d_1, d_2) in R. And so on ...)

Exercise A.16 Consider again the three properties of asymmetry, seriality and transitivity of the previous exercise.

(1) Give a picture of a finite situation with a relation R that is asymmetric and serial but not transitive.

(2) Give a picture of a finite situation with a relation R that is serial and transitive but not asymmetric.

(3) Give a picture of a finite situation with a relation R that is transitive and asymmetric but not serial.

A.5 Functions

Functions are relations with the following special property: for any (a, b) and (a, c) in the relation it has to hold that b and c are equal. Thus a *function* from a set A (called *domain*) to a set B (called *range*) is a relation between A and B such that for each $a \in A$ there is

one and only one associated $b \in B$. In other words, a function is a mechanism that maps an input value to a uniquely determined output value. Looking at the relation *author of* from above, it is immediately clear that it is not a function, because the input *Michael Ende* is not mapped to a unique output but is related to more than one element from the set B of books.

Functions are an important kind of relations, because they allow us to express the concept of *dependence*. For example, we know that the gravitational potential energy of a wrecking ball depends on its mass and the height we elevated it to, and this dependence is most easily expressed in a functional relation.

Functions can be viewed from different angles. On the one hand, they can be seen as sets of data, represented as a collection of pairs of input and output values. This tells us something about the behaviour of a function, i.e. what input is mapped to which output.

The function converting temperatures from Kelvin to Celsius can be seen as a set of pairs $\{(0, -273.15), \ldots\}$, and the function converting temperatures from Celsius to Fahrenheit as a set $\{(-273.15, -459.67), \ldots\}$. Determining the output of the function, given some input, simply corresponds to a table lookup. Any function can be viewed as a – possibly infinite – database table. This is called the extensional view of functions. Another way to look at functions is as *instructions for computation*. This is called the intensional view of functions. In the case of temperature conversion the intensional view is more convenient than the extensional view, for the function mapping Kelvin to Celsius can easily be specified as a simple subtraction

$$x \mapsto x - 273.15$$

This is read as 'an input x is mapped to x minus 273.15'. Similarly, the function from Celsius to Fahrenheit can be given by

$$x \mapsto x \times \frac{9}{5} + 32$$

For example, if we have a temperature of 37 degrees Celsius and want to convert it to Fahrenheit, we replace x by 37 and compute the outcome by multiplying it with $\frac{9}{5}$ and then adding 32.

$$37 \times \frac{9}{5} + 32 \to 66.6 + 32 \to 98.6$$

The example shows that the intensional view of functions can be made precise by representing the function as an expression, and specifying the principles for simplifying (or: rewriting) such functional expressions. Rewriting functional expressions is a form of simplification where part of an expression is replaced by something simpler, until we arrive at an expression that cannot be simplified (or: reduced) any further. This rewriting corresponds to the computation of a function. For example, the function converting Celsius to Fahrenheit applied to the input 37 is the expression $37 \times \frac{9}{5} + 32$. This expression denotes the output, and at the same time it shows how to arrive at this output: First, $37 \times \frac{9}{5}$ is

rewritten to 66.6, according to the rewriting rules for multiplication. The result of this simplification is $66.6 + 32$, which is then rewritten to 98.6, in accordance with the rewriting rules for addition.

Functions can be composed, as follows. Let g be the function that converts from Kelvin to Celsius, and let f be the function that converts from Celsius to Fahrenheit. Then $f \cdot g$ is the function that converts from Kelvin to Fahrenheit, and that works as follows. First convert from Kelvin to Celsius, then take the result and convert this to Fahrenheit. It should be clear from this explanation that $f \cdot g$ is defined by

$$x \mapsto f(g(x)),$$

which corresponds to

$$x \mapsto (x - 273.15) \times \frac{9}{5} + 32$$

Exercise A.17 The successor function $s : \mathbb{N} \to \mathbb{N}$ on the natural numbers is given by $n \mapsto n + 1$. What is the composition of s with itself?

A special function which is simple yet very useful is the *characteristic function* of a set. The characteristic function of subset A of some universe (or: domain) U is a function that maps all members of A to the truth-value **True** and all elements of U that are not members of A to **False**. E.g. the function representing the property of being divisible by 3, on the domain of integers, would map the numbers

$$\ldots, -9, -6, -3, 0, 3, 6, 9, \ldots$$

to **True**, and all other integers to **False**. Characteristic functions characterize membership of a set. Since we specified relations as sets, this means we can represent every relation as a characteristic function.

Exercise A.18 $\leq$ is a binary relation on the natural numbers. What is the corresponding characteristic function?

Exercise A.19 Let $f\colon A \to B$ be a function. Show that the relation $R \subseteq A^2$ given by $(x, y) \in R$ if and only if $f(x) = f(y)$ is an equivalence relation (reflexive, transitive and symmetric) on A.

A.6 Recursion and Induction

A recursive definition is a recipe for constructing objects from a finite number of ingredients in a finite number of ways. An example is the following recursive definition of natural numbers:

- 0 is a natural number.

- adding 1 to a natural number n gives a new natural number $n + 1$.

- nothing else is a natural number.

This recipe gives rise to an important method for proving things: proof by *mathematical induction*.

As an example, we prove the fact about natural numbers that the sum of the first n odd natural numbers equals n^2. For example $1 + 3 = 2^2$, $1 + 3 + 5 + 7 = 4^2$, and so on. More formally and generally, we have for all natural numbers n:

$$\sum_{k=0}^{n-1}(2k + 1) = n^2.$$

Here is a proof of this fact by mathematical induction.

Basis. For $n = 0$, we have $\Sigma_{k=0}^{0}(2k + 1) = 1 = 1^2$, so for this case the statement holds.

Induction step. We assume the statement holds for some particular natural number n and we show that it also holds for $n+1$. So assume $\sum_{k=0}^{n-1}(2k+1) = n^2$. This is the **induction hypothesis**. We have to show: $\sum_{k=0}^{n}(2k + 1) = (n + 1)^2$. Indeed,

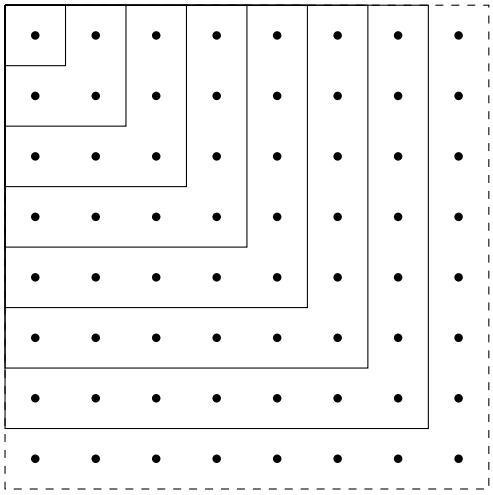
$$\sum_{k=0}^{n}(2k + 1) = \sum_{k=0}^{n-1}(2k + 1) + 2n + 1.$$

Now use the induction hypothesis to see that this is equal to $n^2 + 2n + 1$, which in turn equals $(n + 1)^2$. Therefore we have:

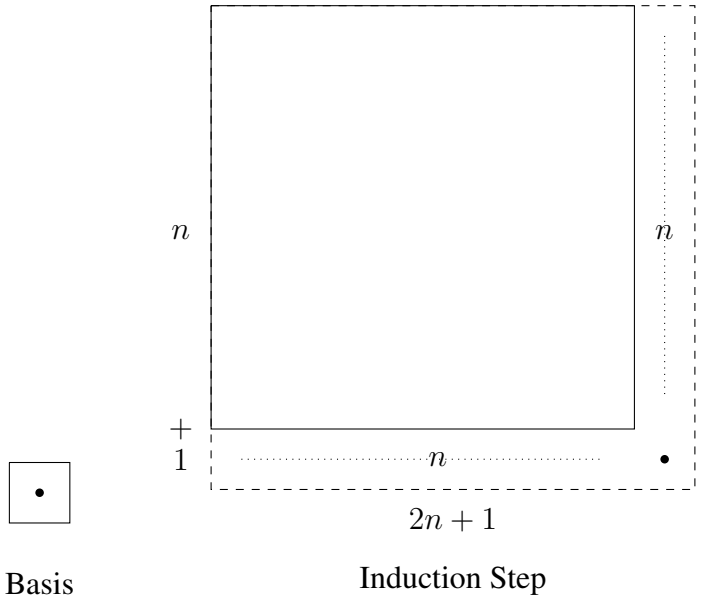
$$\sum_{k=0}^{n}(2k + 1) = \sum_{k=0}^{n-1}(2k + 1) + 2n + 1 \stackrel{ih}{=} n^2 + 2n + 1 = (n + 1)^2.$$

The equality $\stackrel{ih}{=}$ is the step where the induction hypothesis was used. We have checked two cases: the case 0 and the case $n + 1$. By the recursive definition of natural numbers, we have covered *all* cases, for these are the two possible shapes of natural numbers. So we have proved the statement for all natural numbers n.

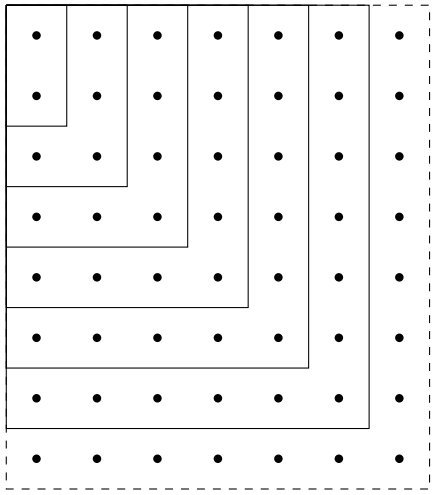
The procedure of proof by mathematical induction does not help to *find* interesting patterns, but once such a pattern is found it is very helpful to *check* whether the pattern really holds. So how can one find a pattern like $\sum_{k=0}^{n-1}(2k + 1) = n^2$ in the first place? By imagining the following way to build up a square with side n:



Such a picture is what the ancient Greeks called a *gnomon* ("thing by which one knows"). The structure of the inductive proof can now be pictured as follows:



Exercise A.20 Consider the following *gnomon*:



What does this suggest for the sum of the first n even numbers? Give a form for $\sum_{k=0}^{n} 2k$, and prove with induction that your form is correct.

Appendix B

Solutions to the Exercises

Solutions to Exercises from Chapter 2

Exercise 2.1 on page 2-7: Consider the case where there are three facts that you are interested in. You wake up, you open your eyes, and you ask yourself three things: "Have I overslept?", "Is it raining?", "Are there traffic jams on the road to work?". To find out about the first question, you have to check your alarm clock, to find about the second you have to look out of the window, and to find out about the third you have to listen to the traffic info on the radio. We can represent these possible facts with three basic propositions, p, q and r, with p expressing "I have overslept", q expressing "It is raining", and r expressing "There are traffic jams." Suppose you know nothing yet about the truth of your three facts. What is the space of possibilities?

$$\left\{ \begin{array}{ll} p\,q\,r & p\,q\,\overline{r} \\ p\,\overline{q}\,r & p\,\overline{q}\,\overline{r} \\ \overline{p}\,q\,r & \overline{p}\,q\,\overline{r} \\ \overline{p}\,\overline{q}\,r & \overline{p}\,\overline{q}\,\overline{r} \end{array} \right\}$$

Exercise 2.2 on page 2-8: (Continued from previous exercise.) Now you check your alarm clock, and find out that you have not overslept. What happens to your space of possibilities?

$$\left\{ \begin{array}{ll} p\,q\,r & p\,q\,\overline{r} \\ p\,\overline{q}\,r & p\,\overline{q}\,\overline{r} \\ \overline{p}\,q\,r & \overline{p}\,q\,\overline{r} \\ \overline{p}\,\overline{q}\,r & \overline{p}\,\overline{q}\,\overline{r} \end{array} \right\} \quad \stackrel{\neg p}{\Longrightarrow} \quad \left\{ \begin{array}{ll} p\,q\,r & p\,q\,\overline{r} \\ p\,\overline{q}\,r & p\,\overline{q}\,\overline{r} \end{array} \right\}$$

Exercise 2.3 on page 2-8: You are given the information that p-or-q and (not-p)-or-r. What is the strongest valid conclusion you can draw?

$$\left\{\begin{array}{l} p\,q\,r \\ p\,q\,\overline{r} \\ p\,\overline{q}\,r \\ p\,\overline{q}\,\overline{r} \\ \overline{p}\,q\,r \\ \overline{p}\,q\,\overline{r} \\ \overline{p}\,\overline{q}\,r \\ \overline{p}\,\overline{q}\,\overline{r} \end{array}\right\} \quad \begin{array}{c} p \vee q \\ \Longrightarrow \end{array} \quad \left\{\begin{array}{l} p\,q\,r \\ p\,q\,\overline{r} \\ p\,\overline{q}\,r \\ p\,\overline{q}\,\overline{r} \\ \overline{p}\,q\,r \\ \overline{p}\,q\,\overline{r} \end{array}\right\} \quad \begin{array}{c} \neg p \vee r \\ \Longrightarrow \end{array} \quad \left\{\begin{array}{l} p\,q\,r \\ p\,\overline{q}\,r \\ \overline{p}\,q\,r \\ \overline{p}\,q\,\overline{r} \end{array}\right\}$$

Any valid conclusion has to be true in the set of remaining alternatives $\left\{\begin{array}{l} p\,q\,r \\ p\,\overline{q}\,r \\ \overline{p}\,q\,r \\ \overline{p}\,q\,\overline{r} \end{array}\right\}$. If it is also false in the set of eliminated alternatives $\left\{\begin{array}{l} \overline{p}\,\overline{q}\,\overline{r} \\ \overline{p}\,\overline{q}\,r \\ p\,\overline{q}\,\overline{r} \\ p\,q\,\overline{r} \end{array}\right\}$ then it is among the strongest ones. For instance $p \vee q$ is a valid conclusion but it is not strong enough because it is also true, for instance, in $p\,q\,\overline{r}$. The formula $(p \vee q) \wedge (\neg p \vee r)$ is among the strongest conclusions that you can draw from the given information (and so is any formula equivalent to it).

Exercise 2.6 on page 2-12:

- I will only go to school if I get a cookie now:

  $$(p \rightarrow q) \wedge (q \rightarrow p)$$

  where p ="I get a cooky now" and q ="I will go to school".

- John and Mary are running:

  $$p \wedge q$$

  where p ="John is running" and q ="Mary is running".

- A foreign national is entitled to social security if he has legal employment or if he has had such less than three years ago, unless he is currently also employed abroad:

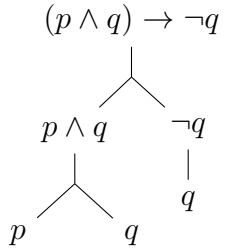
  $$((p \vee q) \wedge \neg r) \rightarrow s$$

  where p ="A foreign national has legal employment", q ="A foreign national has had legal employment less then three years ago", r ="A foreign national is currently also employed abroad" and s ="A foreign national is entitled to social security".

Exercise 2.7 on page 2-12: Only the first one.

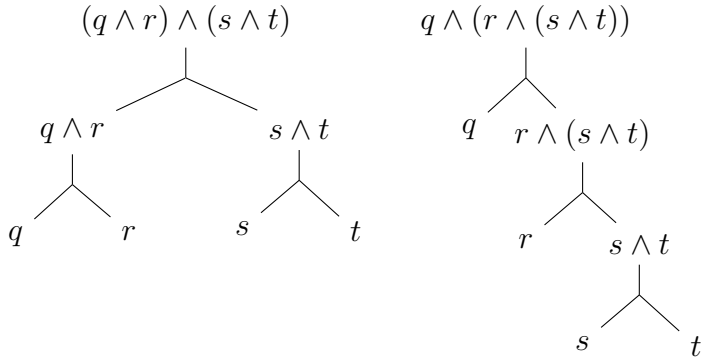
Exercise 2.8 on page 2-12: Construct a tree for the following formulae:

$(p \wedge q) \rightarrow \neg q$:



$q \wedge r \wedge s \wedge t$ (draw all possible trees; and does it matter?)

Two possible trees are depicted below, you should build the remaining ones and check that the order doesn't matter in either construction (in the sense that the logical meaning of this particular formula is invariant under different construction orders). This is not, however, a general result: sometimes the order in the construction tree changes the logical meaning (truth value) of composed formulae.



Exercise 2.11 on page 2-17:

$(p$	$\rightarrow$	$q)$	$\vee$	$(q$	$\rightarrow$	$p)$
0	1	0	1	0	1	0
0	1	1	1	1	0	0
1	0	0	1	0	1	1
1	1	1	1	1	1	1

$((p$	$\vee$	$\neg$	$q)$	$\wedge$	$r)$	$\leftrightarrow$	$(\neg$	$(p$	$\wedge$	$r)$	$\vee$	$q)$
0	1	1	0	0	0	0	1	0	0	0	1	0
0	1	1	0	1	1	1	1	0	0	1	1	0
0	0	0	1	0	0	0	1	0	0	0	1	1
0	0	0	1	0	1	0	1	0	0	1	1	1
1	1	1	0	0	0	0	1	1	0	0	1	0
1	1	1	0	1	1	0	0	1	1	1	0	0
1	1	0	1	0	0	0	1	1	0	0	1	1
1	1	0	1	1	1	1	0	1	1	1	1	1

Exercise 2.12 on page 2-17:

Here are two non-equivalent readings:

$$(\neg p \to q) \vee r \qquad \text{and} \qquad \neg(p \to (q \vee r))$$

you should also check the remaining possibilities.

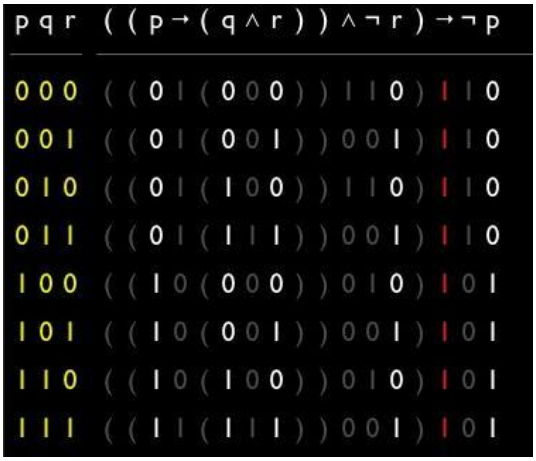
Exercise 2.18 on page 2-20:

p	q	r	$\neg p$	$\neg q$	$q \vee r$	$\neg p \to (q \vee r)$	$p \wedge r$	$p \vee r$
0	0	0	1	1	0	0	0	0
0	0	1	1	1	1	1	0	1
0	1	0	1	0	1	1	0	0
0	1	1	1	0	1	1	0	1
1	0	0	0	1	0	1	0	1
1	0	1	0	1	1	1	1	1
1	1	0	0	0	1	1	0	1
1	1	1	0	0	1	1	1	1

You can check by inspecting the rows of the table that $p \wedge r$ is not a valid consequence and $p \vee r$ is.

Exercise 2.19 on page 2-21:

You can check the first item with the following table:



p q r	((p → (q ∧ r)) ∧ ¬ r) → ¬ p
0 0 0	((0 1 (0 0 0)) 1 1 0) 1 1 0
0 0 1	((0 1 (0 0 1)) 0 0 1) 1 1 0
0 1 0	((0 1 (1 0 0)) 1 1 0) 1 1 0
0 1 1	((0 1 (1 1 1)) 0 0 1) 1 1 0
1 0 0	((1 0 (0 0 0)) 0 1 0) 1 0 1
1 0 1	((1 0 (0 0 1)) 0 0 1) 1 0 1
1 1 0	((1 0 (1 0 0)) 0 1 0) 1 0 1
1 1 1	((1 1 (1 1 1)) 0 0 1) 1 0 1

You might want to use the *Truth Tabulator* applet to build the truth table for the second item. The address is:

`http://staff.science.uva.nl/~jaspars/AUC/apps/javascript/proptab`

Exercise 2.20 on page 2-21:

(a)

$$\left\{ \begin{array}{l} q\,r \\ q\,\overline{r} \\ \overline{q}\,r \\ \overline{q}\,\overline{r} \end{array} \right\} \quad \begin{array}{c} \neg(q \wedge r) \\ \Longrightarrow \end{array} \quad \left\{ \begin{array}{l} q\,\overline{r} \\ \overline{q}\,r \\ \overline{q}\,\overline{r} \end{array} \right\} \quad \begin{array}{c} q \\ \Longrightarrow \end{array} \quad \left\{ \; q\,\overline{r} \; \right\}$$

All the valuations making the premises true also make the conclusion true. We can also see that updating with the conclusion is redundant.

$$\left\{ \begin{array}{l} p\,q\,r \\ p\,q\,\overline{r} \\ p\,\overline{q}\,r \\ p\,\overline{q}\,\overline{r} \\ \overline{p}\,q\,r \\ \overline{p}\,q\,\overline{r} \\ \overline{p}\,\overline{q}\,r \\ \overline{p}\,\overline{q}\,\overline{r} \end{array} \right\} \quad \begin{array}{c} \neg p \vee \neg q \vee r \\ \Longrightarrow \end{array} \quad \left\{ \begin{array}{l} p\,q\,r \\ p\,\overline{q}\,r \\ p\,\overline{q}\,\overline{r} \\ \overline{p}\,q\,r \\ \overline{p}\,q\,\overline{r} \\ \overline{p}\,\overline{q}\,r \\ \overline{p}\,\overline{q}\,\overline{r} \end{array} \right\} \quad \begin{array}{c} q \vee r \\ \Longrightarrow \end{array} \quad \left\{ \begin{array}{l} p\,q\,r \\ p\,\overline{q}\,r \\ \overline{p}\,q\,r \\ \overline{p}\,q\,\overline{r} \\ \overline{p}\,\overline{q}\,r \end{array} \right\} \quad \begin{array}{c} p \\ \Longrightarrow \end{array} \quad \left\{ \begin{array}{l} p\,q\,r \\ p\,\overline{q}\,r \end{array} \right\}$$

We can see that updating with the conclusion has no further effect, hence the consequence relation is valid.

Exercise 2.22 on page 2-21: By comparing the first and the second tables below you can determine that the equivalence does not hold for item (3) while by comparing the first and the third tables below you can determine that the equivalence does hold for item (4). The other items are similar.

$\neg$	$(p$	$\rightarrow$	$q)$
0	0	1	0
0	0	1	1
1	1	0	0
0	1	1	1

p	$\vee$	$\neg$	q
0	1	1	0
0	0	0	1
1	1	1	0
1	1	0	1

p	$\wedge$	$\neg$	q
0	0	1	0
0	0	0	1
1	1	1	0
1	0	0	1

Exercise 2.26 on page 2-27: Disjunction can be defined as:

$$\varphi \vee \psi \equiv \neg(\neg\varphi \wedge \neg\psi)$$

Implication can be defined as:

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$

after which you can use the previous definition for disjunction.

Equivalence can be defined as:

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

and then get rid of implication by the previous definition.

Exclusive disjunction can be defined as:

$$\varphi \oplus \psi \equiv \neg(\varphi \leftrightarrow \psi)$$

and then unfold the definition of equivalence.

The Sheffer stroke can be defined as:

$$\varphi \mid \psi \equiv \neg(\varphi \wedge \psi)$$

and you can continue in this way (see also exercise 2.29 on page 2-36).

Exercise 2.29 on page 2-36:

The following table shows how starting from p and q we can obtain new truth functions applying $\mid$ to previous combinations.

		s_1:	s_2:	s_3:	s_4:	s_5:	s_6:	s_7:	s_8:	s_9:
p	q	$p \mid p$	$p \mid q$	$q \mid q$	$p \mid s_1$	$p \mid s_2$	$q \mid s_1$	$s_1 \mid s_3$	$s_2 \mid s_4$	$s_2 \mid s_7$
1	1	0	0	0	1	1	1	1	1	1
1	0	0	1	1	1	0	1	1	0	0
0	1	1	1	0	1	1	0	1	0	0
0	0	1	1	1	1	1	1	0	0	1

Note that there is no unique way these combinations can be obtained, for instance, s_8 could have been obtained also as $s_2 \mid s_2$. Note that $\varphi \mid \varphi$ defines $\neg\varphi$. Using this observation, the remaining five truth functions can be obtained as negations: for instance, $s_{10} = s_9 \mid s_9$.

Exercise 2.30 on page 2-36: In three ways: (1) yes,no,yes; (2) yes,no,no; (3) no,yes,no.

Exercise 2.31 on page 2-36:

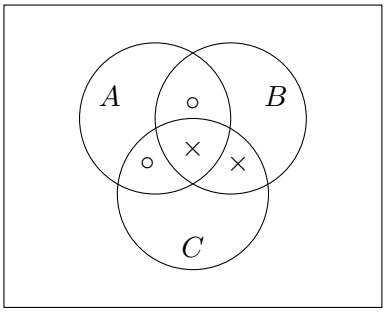
(1) You have to fill in 11 entries.

(2) In the worst case, you have to fill 88 cells in the truth table, which is eight times the number of logical symbols.

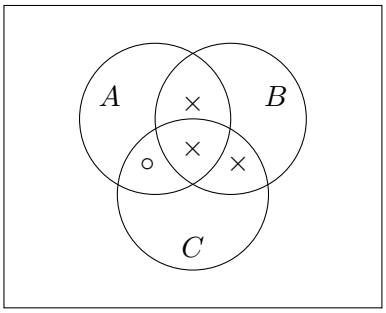
Solutions to Exercises from Chapter 3

Exercise 3.1 on page 3-5: The middle term is B.

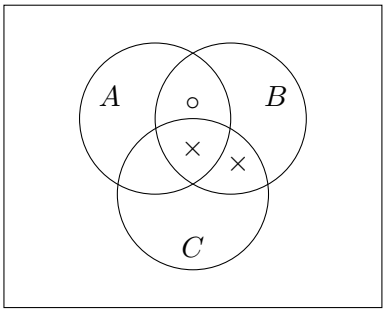
Exercise 3.2 on page 3-15: The syllogistic pattern is not valid because it is possible to build a counter-example in which the premises are both true but the conclusion is false. This can be seen in the following diagram, where A = 'philosopher', B = 'barbarian' and C = 'greek':



Exercise 3.3 on page 3-16: The syllogistic pattern is not valid because it is possible to build a counter-example in which the premises are both true but the conclusion is false. This can be seen in the following diagram, where A = 'philosopher', B = 'barbarian' and C = 'greek':

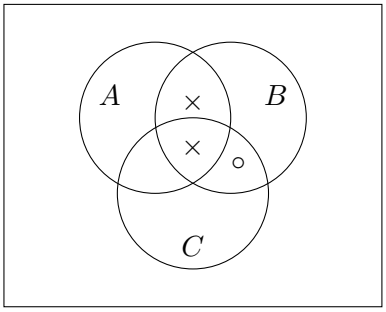


Exercise 3.4 on page 3-16: The syllogistic pattern is valid because after we update with the information given in the premises it is impossible for the conclusion to be false. This can be illustated by the following diagram, where A = 'philosopher', B = 'barbarian' and C = 'greek':



Exercise 3.5 on page 3-16: To obtain the required modification of the method for checking syllogistic validity with the all quantifiers read with existential import we have to represent explicitly the implicit assumption that we are not thinking with empty terms. In this way even universally quantified sentences, which give empty regions in the diagram, might also implicitly give the information that another region is not empty (otherwise some of the terms used in the reasoning will become empty, agains the assumption). To do this in a systematic manner we have to add three existential sentences to the given premises: one for each of the terms in the syllogism. This will correspond to putting a $\circ$ symbol in each circle representing a term while respecting the other premisses. Only after this step will our method further proceed towards checking the effects of updating with the information given in the conclusion. In this way, the following example turns out to be a valid patern of syllogistic reasoning: 'All men are mortal, all greeks are men, therefore, some greeks are mortal'. (This syllogistic mode, mnemotechnically called Barbari, was considered valid by Aristotle as it is the subordinate mode of Barbara.) In general, the inference from 'All A are B' to 'Some A are B' is valid under existential import.

Exercise 3.6 on page 3-21: (1) The first syllogism is invalid and the second one is valid. (2) The following diagram illustrates the validity of the right syllogism:



An alternative representation that makes the stages in the update explicit is the following:

	A	B	C
AB	AC	BC	ABC

$$\Downarrow \neg p_{AB} \wedge \neg p_{ABC}$$

	A	B	C
× AB	AC	BC	× ABC

$$\Downarrow p_{BC} \vee p_{ABC}$$

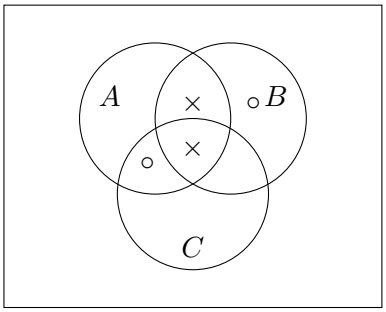
	A	∘ B	∘ C
× AB	AC	∘ BC	× ABC

$$\Downarrow p_{BC} \vee p_{C}$$

	A	∘ B	∘ C
× AB	AC	∘ BC	× ABC

We can see that updating with the conclusion does not add any new information that was not already present in the premises, hence the syllogism is valid.

(3) The following diagram ilustrates how a counterexample to the left syllogism can be constructed:



An alternative way to represent the process of finding a counterexample lists the update at each stage of the reasoning process:

	A	B	C
AB	AC	BC	ABC

$$\Downarrow \neg p_{AB} \wedge \neg p_{ABC}$$

	A	B	C
× AB	AC	BC	× ABC

$$\Downarrow p_{B} \vee p_{AB}$$

	A	∘ B	C
× AB	AC	BC	× ABC

$$\Downarrow p_{ABC} \vee p_{AC}$$

	∘ A	∘ B	∘ C
× AB	∘ AC	BC	× ABC

We can see that updating with the conclusion adds some new information which was not already contained in the information from the premisses, hence the inference amplifies the infomation and makes the syllogisatic reasoning invalid.

(4) For the left syllogism we have:

The initial space of valuations is a state of ignorance about the 8 propositional valuations describing the regions of a Venn diagram, this is a big number: $2^8 = 256$. Using blanks to condense the set gives us a very simple representation:

$p_\emptyset$	p_A	p_B	p_C	p_{AB}	p_{AC}	p_{BC}	p_{ABC}

After the update with the information in the first premise this space of posible valuation is $2^6 = 64$. This is 4 times smaller than before, but still to large to fit on a A4 page, unless we use abbreviated notation:

$p_\emptyset$	p_A	p_B	p_C	p_{AB}	p_{AC}	p_{BC}	p_{ABC}
				0			0

After the update with the second premise the space of valuations is halved, so now we have $2^5 = 32$ possible valuations. Again, it makes sense to abbreviate. Here is the condensed version of the list of possibilities at this stage:

$p_\emptyset$	p_A	p_B	p_C	p_{AB}	p_{AC}	p_{BC}	p_{ABC}
		1		0			0

To check whether the conclusion holds we have to check whether $p_{ABC} \vee p_{AC}$ holds in all these valuations. This is not the case, hence the inference is invalid.

For the right syllogism the initial space of possible valuations is also the state of ignorance about the 8 propositions describing the regions of a Venn diagram. The condensed version of this list of valuations is the same as before. After the first update, with $\neg p_{AB} \wedge \neg p_{ABC}$, we get:

$p_\emptyset$	p_A	p_B	p_C	p_{AB}	p_{AC}	p_{BC}	p_{ABC}
				0			0

After the second update, with $p_{BC} \vee p_{ABC}$, we get:

$p_\emptyset$	p_A	p_B	p_C	p_{AB}	p_{AC}	p_{BC}	p_{ABC}
				0		1	0

All of the valuations represented here make $p_{BC} \vee p_C$ true, hence the argument is valid.

Solutions to Exercises from Chapter 4

Exercise 4.1 on page 4-4: We may consider a situation where both x and y are small, or neither x nor y is small according to the property "small". But it is still possible that x is smaller than y or y is smaller than x, which may not be expressed well by the unary predicate "small" only. Even with help of the notion context-dependent, such as "small compared to", it is not adequate either. For example, within the domain of nature numbers, both 4 and 5 are small compared to 6 but 4 is smaller than 5. One may argue, it is possible to say that 4 is small compared to 5 but 5 is not small compared to itself, then we can combine these two statements to express "4 is smaller than 5". However, it can be seen that, when presented with two different numbers, we always need to use the big one as a compared parameter. It seems ad hoc (every expression needs a different context) and actually shows that "smaller" is fundamentally binary.

Exercise 4.2 on page 4-5: $\neg(x < y < z)$ is an abbreviation of $\neg(x < y \wedge y < z)$. So this works out as $\neg x < y \vee \neg y < z$.

Exercise 4.3 on page 4-6: We use L for "love", j for "John", m for "Mary", and p for "Peter" in the above context, then we may translate the four sentences into the following first order formulas: (1) $Ljm \rightarrow Lmj$, (2) $Ljm \wedge Lmj$, (3) $\neg(Ljm \wedge Lmj)$, (4) $(Ljm \wedge Lpm) \rightarrow (\neg Lpj \wedge \neg Lmj)$.

Exercise 4.4 on page 4-6: $(x < y \vee x = y) \wedge (y < z \vee y = z)$

Exercise 4.5 on page 4-6: $\neg((x < y \vee x = y) \wedge (y < z \vee y = z))$ or $\neg(x < y \vee x = y) \vee \neg(y < z \vee y = z)$

Exercise 4.6 on page 4-7: $\forall x(Bx \wedge Wx)$ expresses "All x have the B and W properties" or "Everything is a boy and walks". $\exists x(Bx \rightarrow Wx)$ expresses "There is an x such that if he is B then he has the W property" or "There is something (someone) that walks if it's a boy".

Exercise 4.7 on page 4-9: (1) $\forall x(Bx \rightarrow \neg Cx)$ or $\neg\exists x(Bx \wedge Cx)$ (2) $\exists x(Bx \wedge Cx)$ (3) $\neg\forall x(Ax \rightarrow Bx)$

Exercise 4.8 on page 4-12: (1) $\exists x \neg Lxa$, (2) $Ral \wedge Rla$, (3) $\forall x(Lxa \rightarrow Rlx)$.

Exercise 4.9 on page 4-12: $\exists x(Bx \wedge \neg Fx)$

Exercise 4.10 on page 4-12:

(1) Domain of discourse: all dogs, key: B for "barking", W for "biting", translation: $\forall x \neg (Bx \wedge Wx)$,

(2) Domain of discourse: all inaminate objects, key: G for "glittering", A for "being gold", translation: $\exists x(Gx \wedge \neg Ax)$,

(3) Domain of discourse: all human beings, key: F for "friendship", m for Michelle, translation: $\forall xy((Fmx \wedge Fxy) \rightarrow Fmy)$,

(4) Domain of discourse: the set of natural numbers $\mathbb{N}$, key: S for "smaller than" , translation: $\exists x \forall y Sxy$,

(5) Domain of discourse: the set of natural numbers $\mathbb{N}$, key: P for "being prime", S for "smaller than", translation: $\forall x(Px \rightarrow \exists y(Py \wedge Sxy))$.

Exercise 4.11 on page 4-12:

(1) $\forall x(Bx \rightarrow Lxm)$,

(2) $\exists x(Gx \wedge \neg Lxx)$,

(3) $\neg \exists x((Bx \vee Gx) \wedge Lxp)$,

(4) $\exists x(Gx \wedge Lpx \wedge Lxj)$.

Exercise 4.12 on page 4-13:

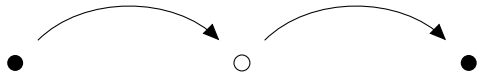
(1) 

where solid dots are boys, open dots are girls, the arrow represents the love relation, and Mary is the right dot.

(2) the same picture as for item (1).

(3) the same picture as for item (1), with Peter the left dot.

(4) the following picture, with John the dot on the right:



Exercise 4.13 on page 4-13:

(1) $\exists x \exists y(Bx \wedge Gy \wedge \neg Lxy)$,

(2) $\forall x((Bx \wedge \exists y(Gy \wedge Lxy)) \rightarrow \exists z(Gz \wedge Lzx))$,

(3) $\forall x((Gx \wedge \forall y(By \rightarrow Lxy)) \rightarrow \exists z(Gz \wedge \neg Lxz))$,

(4) $\forall x \forall y(((Gx \wedge \forall v(Bv \rightarrow \neg Lxv)) \wedge (Gy \wedge \exists z(Bz \wedge Lyz))) \rightarrow \neg Lxy)$.

Exercise 4.14 on page 4-14: (1) $\neg\forall x\neg(Ax \wedge Bx)$ (2) $\neg\forall x(Ax \rightarrow Bx)$ (3) $\forall x(Ax \rightarrow \neg Bx)$

Exercise 4.15 on page 4-14: (1) $\neg\exists x(Ax \wedge \neg Bx)$ (2) $\neg\exists x(Ax \wedge Bx)$ (3) $\neg\exists x(Ax \wedge Bx)$

Exercise 4.18 on page 4-17:

(1) $\forall x(Gx \rightarrow \exists y(Hy \wedge Wxy)$

(2) $\forall x \forall y(Gx \wedge Hy \rightarrow Wxy)$ or $\exists x(Gx \wedge \forall y(Hy \rightarrow Wxy)$.

Exercise 4.19 on page 4-20:

(1) Yes,

(2) No,

(3) Yes.

Exercise 4.21 on page 4-20:

(1) No. We may find two different people *a*, *b*, with neither *a* an ancestor of *b* nor *b* an ancestor of *a*.

(2) No. We may find two different people *c*, *d*, with neither *c* a parent of *d* nor *d* a parent of *c*.

(3) Yes. For any two natural numbers *m* and *n* it holds that either *m* is less than *n* or *n* is less than *m*, or they are equal.

Exercise 4.22 on page 4-21. Let t represent Tutankhamun. Then the formula has to express that the parents of Tutankhamun's mother are also the parents of Tutankhamun's father. In the following formula, x, y are father and mother of Tutankhamun, and u, v are grandfather and grandmother. Tutankhamun has only one grandfather and only one grandmother.

$$\exists x \exists y(Mx \wedge \neg My \wedge Pxt \wedge Pyt \wedge \exists u \exists v(Mu \wedge \neg Mv \wedge Pux \wedge Puy \wedge Pvx \wedge Pvy)).$$

Exercise 4.24 on page 4-22: $\exists x \exists y \exists z(Rxy \wedge Ryz)$ is true in the lefthand picture, but false in the righthand picture.

Exercise 4.25 on page 4-23: Just add a single $\rightarrow$ loop to the middle point of the graph, meaning that the middle point has an *R* relation with itself.

Exercise 4.26 on page 4-23: Let A be a finite set and let R be a relation on A that is reflexive and connected. Connectedness is expressed as $\forall x \forall y(Rxy \vee Ryx)$. A great communicator c is a point with the following property:

$$\forall x(Rcx \vee \exists y(Rcy \wedge Ryx)).$$

We show that each finite A has such a c, as follows. If A has just a single point, that point must be c, by reflexivity. If A has more than one point, then by removing a point x from A we get a smaller set B. Assume this B has a communicator c. By connectedness, either cRx or xRc. In the first case, c is a communicator for A. In the second case, if there is an R-successor y of c in B with yRx, then x can be reached from c in two steps, so again c is a communicator for A. On the other hand, if no R-successor of c in B has an R link to x, then by connectedness, there must be an R link from x to every R-successor of c. But this means that x is a communicator for A.

This induction argument shows that the property holds for any finite reflexive and connected set. The property does not hold for infinite sets. Consider $\mathbb{N}$ with R interpreted as $\geq$. Then R is reflexive and connected, but there is no great communicator, for a great communicator would be a number that is at least as large as any other number.

Exercise 4.27 on page 4-27: The occurrences of x in Rxy and $Sxyz$.

Exercise 4.28 on page 4-27: $\exists x$ binds the occurrences of x in Rxx and $\forall x$ binds the occurrence of x in Px.

Exercise 4.29 on page 4-27: (1) and (5).

Exercise 4.30 on page 4-28: (2) and (4). They are equivalent to $\forall x Rxx$ and $\exists y Ryy$ respectively.

Exercise 4.31 on page 4-29: (1) Rcc, (2) Ryy, (3) $\forall x Rxx$, (4)$\forall y Ryy$, (5) $\exists y Rzy$.

Exercise 4.32 on page 4-35: (1), (2), (4), (6), (9).

Exercise 4.33 on page 4-36: (1) Holds because we assume nonempty domains, (2) Doesn't hold: $D = \{1, 2\}, I(P) = \{1\}$, (3) Holds because we can chose the same object twice, (4) Doesn't hold: $D = \{1, 2\}, I(R) = \{(1, 2), (2, 1)\}$, (5) Doesn't hold: $D = \{1, 2\}, I(R) = \{(1, 2)\}$, (6) Doesn't hold: $D = \{1, 2\}, I(R) = \{(1, 2), (2, 2)\}$, (7) Holds because we can reuse the choice for y in the premise when we chose again in the conclusion, (8) Doesn't hold: $D = \{1, 2\}, I(R) = \{(1, 2), (2, 1)\}$, (9) Doesn't hold: $D = \{1, 2\}, I(R) = \{(1, 2)\}$, (10) see point (3), (11) Holds because the same object can be chosen for both x and y in the conclusion.

Exercise 4.34 on page 4-37: (1) Holds, (5) Holds, (6) Holds.

Exercise 4.35 on page 4-40: Assume that $A = A_1, A_2, ...A_n, \ldots$ is an enumeration of the valid

formulas of the language, and $B = B_1, B_2, ...B_n, \ldots$ is an enumeration of the formulas that are not valid. Let φ be an arbitrary formula of the language. Observe that either φ is valid or it is not. Therefore, either φ occurs in the first sequence or it occurs in the second sequence. The following procedure is a decision method: If φ equals A_0 then φ is valid, if φ equals B_0 then φ is not valid. And so on: if φ equals A_i then φ is valid, if φ equals B_i then φ is not valid. Since φ is either valid or not valid, there must be an i for which $\varphi = A_i$ or $\varphi = B_i$.

Exercise 4.36 on page 4-43:

$\exists x \exists y \exists z \left(\neg x = y \wedge \neg x = z \wedge \neg x = y \wedge \forall v \left(Pv \leftrightarrow (v = y \vee v = x \vee v = z)\right)\right)$

Exercise 4.37 on page 4-43: $P(x) \leftrightarrow \exists!^2 y(y|x)$ where $x, y \in \mathbb{N}$. When we replace $\exists!^2$ in this definition by $\exists!^3$ we get numbers that have exactly three divisors. These numbers have to be squares of primes: if p is a prime number, then p^2 has $\{1, p, p^2\}$ as its set of divisors. Conversely, if n has exactly three divisors, it has to be of this form.

Exercise 4.38 on page 4-43:

(1):

(2) in a model with two elements one of which is A but not B and the other is B but not A the formulas have a different truth value: $\exists!x \left(Ax \vee Bx\right)$ is false and $\exists!x\, Ax \vee \exists!x\, Bx$ is true.

Exercise 4.39 on page 4-48:

$$\exists y\ y + y + y = x. \qquad (x \text{ is a threefold})$$

Exercise 4.40 on page 4-50: Consider what the function would do for input 4. This input satisfies the precondition, so according to the contract the result should be $\texttt{ld}(4) = 2$. But this is not what we get. The procedure starts by assigning 2 to `d`. Next, the check `d**2 < n` is performed. This check fails, for $d^2 = 2^2 = 4$. Therefore, the while loop will not be executed, and the function returns the value of `n`, i.e., the function returns 4.

Exercise 4.16 on page 4-15: $\exists$ distributes over $\vee$, but it does not hold for $\wedge$. Lets confine our domain in the set of natural numbers. We may consider a situation in which $\exists x Ex \wedge \exists x Ox$ holds, (say that there exists an even number and there exists an odd number) but $\exists x(Ex \wedge Ox)$ does not necessarily hold (there is no natural number which is both even and odd).

Exercise 4.17 on page 4-16: You may see other possible implications between repeated quantifiers as follows

(1) $\exists x \exists y \varphi \rightarrow \exists y \exists x \varphi$, valid

(2) $\exists x \forall y \varphi \to \forall y \forall x \varphi$, not valid

(3) $\forall x \forall y \varphi \to \exists y \forall x \varphi$, not valid

(4) $\forall x \forall y \varphi \to \exists y \exists x \varphi$, not valid

(5) $\forall x \forall y \varphi \to \forall y \exists x \varphi$, not valid

(6) $\forall x \exists y \varphi \to \exists y \exists x \varphi$, not valid

If we assume that domains are non-empty, then the last four implications (but not their converses) will become valid.

Solutions to Exercises from Chapter 5

Exercise 5.2 on page 5-3: For players in a soccer match, they can observe what's going on, communicate with each other mainly in their self team, make inferences by themselves, and etcetera. The observation channel may be restricted by players from enemy team attempted and by players from self team unconsciously. For the details of those main three channels, you may see the following explanation: A system to play soccer consists of 22 agents, the players. A soccer player can only observe the players in his range of vision. This determines what they know and believe, and how they can obtain more information. Player a assumes that opponent b is behind him, because in the previous stage of the game, before a received the ball, this was indeed the case. This is therefore a reasonable assumption, and a also knows that such an assumption can very well have become false. It turns out to be false: b is by now somewhere else on the playing ground, but a could not have seen that. The state of the game where b is behind a, is just as conceivable for a as the state of the game where b is not behind a. Player a also sees player c right in front of himself. Player c is in his own team; a passes the ball to c to prevent b from intercepting it. Now, what? The targeted player indeed gets the ball, only it was player d instead of c, fortunately of the same team, who received the ball. Player a *believed* player d to be c, but must now revise the previous assumption he made. In yet another scenario c fails to capture the ball, because a and c do not have eye-contact at the moment of a's pass: they do not have common knowledge of the situation leading to this pass.

Exercise 5.8 on page 5-8: You uploaded a message p in your website. Then I logged on (visited) your website with my registered account and see the message p. I also found my visiting trace appeared in your website and you logged on as well after that, and I know that you can figure out the identities of visitors by account names according to the registered information. Then we may say, I know that you know that I know p.

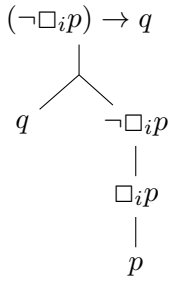
Exercise 5.14 on page 5-12: For example, teachers often know answers but they may ask students questions in order to improve learning efficiency for students. As in the card example, if 2 has a blue card and asks 1 "do you have the blue card?", then 1 answers "No" truthfully. After that, 3 knows the distribution of cards and knows that 2 asked 1 a misleading question. 1 is not certain if the blue card is in 2 or 3 for she cannot decide whether 2 has asked her a misleading question. 2

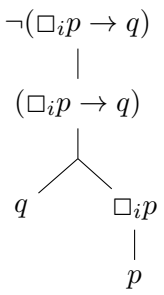
knows that she asked 1 a misleading question but she still cannot figure out the actual distribution of cards from the answer of 1. However 2 does know that 3 knows the distribution of cards and she asked a misleading question.

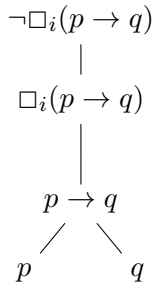
Exercise 5.17 on page 5-13: We may have the following correct formulas by putting in brackets at appropriate places.

$$(\neg\Box_i p) \to q, \neg(\Box_i p \to q), \neg\Box_i(p \to q)$$

Their corresponding analysis trees are as follows







Exercise 5.19 on page 5-14: In classes, teachers often know the answer of a question they ask students, and they don't expect that every student of answerers may know the answer.

Exercise 5.20 on page 5-14: We use p, j, m, to represent 'it is raining', John, and Marry respectively.

(1) $\Box_j \neg p$

(2) $\Box_j(\Box_m p \vee \Box_m \neg p)$

(3) $\Box_j((\Box_m p \vee \Box_m \neg p) \vee ((\neg \Box_m p) \wedge (\neg \Box_m \neg p)))$

Exercise 5.21 on page 5-15:

(1) 1 knows that if p is the case, then 2 does not know q.

(2) If 1 knows that 2 knows that p, then 2 knows that 1 knows that p.

Exercise 5.28 on page 5-19: First we check that $(p \rightarrow \Box_a p) \wedge (\neg p \rightarrow \Box_a \neg p)$ is true at all states of the model. It is clear that $(p \rightarrow \Box_a p)$ is true in s_0 and s_2 for p is false in those two states. And in s_1 and s_3, it is easy to see that $\Box_a p$ is true. So we have $(p \rightarrow \Box_a p)$ is true in all those four states of the model. Similarly, $\neg p \rightarrow \Box_a \neg p)$ is true in s_1 and s_3 since p is true there. And it's easy to see that $\Box_a \neg p$ is true in s_0 and s_2. So the right conjunct of the original formula is also true in all those four states of model. Then we can conclude that the conjunction is true in all the states of the model. Similarly we can check the second formula that express 'b knows its own state q' is also valid in all the states of the model as well.

Exercise 5.29 on page 5-20: Let s be an arbitrary state of the model and suppose $\Diamond_a \Box_b \varphi$ is true there. Then there exists a state t which is accessible from s via process a and $\Box_b \varphi$ holds in t. But if state t has a b-path leading to a state called u, then φ is true there. By the structure of the model, there is also a state which connects u via the process a, noted as w. Then we have $mK_a \varphi$ holds in w. But w is just the state which is accessible from the state s via the process b by the construction of the model. And there is only one state which can be accessed by s via process b. Hence, $\Box_b \Diamond_a \varphi$ holds in state s and so does $\Diamond_a \Box_b \varphi \rightarrow \Box_b \Diamond_a \varphi$. Since s is arbitrary, this formula is true in all states of the model.

Exercise 5.30 on page 5-20:

(1) $\Diamond t$ is true at w_6 and w_8,

(2) $\Diamond \Box t$ is true in w_3, w_5, w_6, w_7 and w_8,

(3) $\Diamond p$ is true in w_2, w_3, w_4 and w_5,

(4) $\Box \Diamond p$ is true in w_1 and w_2.

Next, for each world, we may find an respective epistemic formula which is only true at that state.

(1) w_9: $\Box p \wedge \Box t \wedge t$,

(2) w_8: $\Box \neg p \wedge \Box t$,

(3) w_7: $\Box \Box \neg p \wedge \Box \Box t$,

(4) w_6: $\Box \neg p \wedge \Box t \wedge p$,

(5) w_5: $\Diamond p \wedge \Box \Box t \wedge \Diamond \neg p$,

(6) w_4: $\Diamond\Box\Box\neg p \wedge \Diamond\Box\Box t)$,

(7) w_3: $\Box p \wedge \Box\Box t \wedge \neg t$,

(8) w_2: $\Diamond p \wedge \Diamond\Diamond p \wedge \Diamond\Box\Box t$

(9) w_1: $\Diamond(\Diamond\Box\Box\neg p \wedge \Diamond\Box\Box t) \wedge \Diamond(\Diamond p \wedge \Diamond\Diamond p \wedge \Diamond\Box\Box t)$.

Exercise 5.31 on page 5-20:

(1) For the world w: $\Box_2 p \wedge \Box_1 p \wedge \Box_1\Box_2 p \wedge \neg\Box_2\Box_1 p$

(2) For the world v: $\Box_2 p \wedge \neg\Box_1 p \wedge \neg\Box_1\neg p$

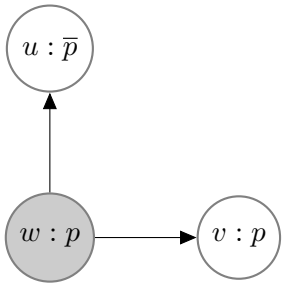
(3) For the world u: $\Box_1 p \wedge \Box_2 p \wedge \Box_1\Box_2 p \wedge \Box_2\Box_1 p$

(4) For the world s: $\Box_2\neg p \wedge \neg\Box_1 p \wedge \neg\Box_1\neg p$

Exercise 5.32 on page 5-21: We know the final model is as follows:

Let p, q, r represent '1 owns red card', '2 owns white card', '3 owns blue card' respectively. In the actual situation and actually all the situations of the final model , we can verify that $\Box_1(p \wedge q \wedge r) \wedge \Box_2(p \wedge q \wedge r)$ (1 and 2 know what the deal is), and $\neg\Box_3 p \wedge \neg\Box_3 q \wedge \Box_3 r$ (3 does not know what the deal is), but $\Box_3(\Box_1(p \wedge q \wedge r) \wedge \Box_2(p \wedge q \wedge r))$ (3 does know that 1 and 2 know the distribution of the cards).

Exercise 5.36 on page 5-24:



Note that in the actual world of the model $\neg\Box p$ is true but $\Box\neg\Box p$ is false since $\Box p$ is true in the world v which is accesible from actual world w. Thus, the formula of negative introspection does not hold in the model.

Exercise 5.37 on page 5-24: Suppose a, b, c are arbitrary worlds satisfying aRb and aRc. Since R is symmetric, we have bRa. Then combined with aRc, it is easy to get bRc (as required) by the transitivity of R.

Exercise 5.38 on page 5-24: Suppose that a, b are arbitrary worlds satisfying aRb. We need to show bRa. Since R is reflexive, it's easy to have aRa. From aRb and aRa, we can conclude bRa by the Euclidic property of R.

Exercise 5.39 on page 5-24: The first one is valid. Suppose $\Diamond_1\Box_2\varphi$ is true in an arbitrary world w of a model with equivalence relations. Then there exists a world u such that wR_1u and $\Box_2\varphi$ is true in u. Since R_2 is reflexive (by equivalence relations), we get φ is also true in u. It's then easy to have '$\Diamond_1\varphi$ is true in w' since wR_1u. And wR_2w (by reflexivity), so we conclude '$\Diamond_2\Diamond_1\varphi$ is true in w', as required. The second is not valid. Please see the following counter-example:

The line between w and v represents 1's equivalence relation only, and other reflexive circles are omitted. It's easy to see that $\Box_2 p$ is true in u. Then we have $\Diamond_1\Box_2 p$ is true in w since wR_1u. But $mK_2\Box_1 p$ does not hold in w since $\neg p$ holds there (actually $\Box_1 p$ is false in both worlds).

Exercise 5.40 on page 5-25: Suppose we have an arbitrary information model with an accessibility relation R that satisfies $\forall x\forall y\forall z((Rxy \wedge Rxz) \rightarrow Ryz)$. Let w be an arbitrary world of the model and $\neg\Box\varphi$ be true in w. We need to show that $\Box\neg\Box\varphi$ is also true in the same world. It's clear to have that $\Diamond\neg\varphi$ is true in w first, and then, we can get 'there exists a world u which is accessible from w and $\neg\varphi$ is true in u' by the semantics of $\Diamond$. If w has no other accessible worlds (besides u), by wRu and the property of R, we have uRu. It shows that $\Diamond\neg\varphi$ is true in u as well. This means $\Box\Diamond\neg\varphi$ is true in w, as required. For the interesting case when w has other accessible worlds different from u, just consider the arbitrary one, say, v, we can conclude that uRv and vRu plus uRu and vRv. Then $\Diamond\neg\varphi$ is also true in v (and u). Since v is arbitrary, it follows that $\Box\Diamond\neg\varphi$ is true in w, as required.

Exercise 5.46 on page 5-28:

Proof.

(1) $\vdash (\neg\psi \wedge (\neg\psi \rightarrow \neg\varphi)) \rightarrow \neg\varphi$ propositional logic

(2) $\vdash \Box(\neg\psi \wedge (\neg\psi \rightarrow \neg\varphi)) \rightarrow \Box\neg\varphi$ distribution rule on 1

(3) $\vdash \Box\neg\psi \wedge \Box(\neg\psi \rightarrow \neg\varphi) \rightarrow \Box\neg\varphi$ example refBoxConjConjBoxes, 2

(4) $\vdash \Box\neg\psi \rightarrow (\Box(\neg\psi \rightarrow \neg\varphi) \rightarrow \Box\neg\varphi)$ propositional logic, 3

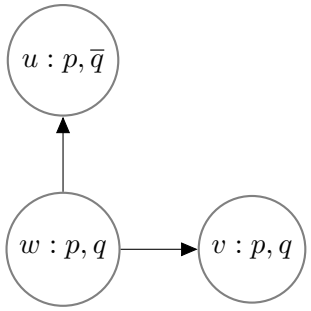
(5) $\vdash \Box\neg\psi \rightarrow (\Box\neg\varphi \vee \Diamond\neg(\neg\psi \rightarrow \neg\varphi))$ propositional logic and definition of $\Diamond$, 4

(6) $\vdash \neg(\Box\neg\varphi \vee \Diamond\neg(\neg\psi \rightarrow \neg\varphi) \rightarrow \neg\Box\neg\psi$ propositional logic, 5

(7) $\vdash (\neg\Box\neg\varphi \wedge \neg\Diamond\neg(\varphi \rightarrow \psi)) \rightarrow \neg\Box\neg\psi$ propositional logic, 6

(8) $\vdash \Diamond\varphi \wedge \Box(\varphi \rightarrow \psi)) \rightarrow \Diamond\psi$ definition of $\Diamond$, 7

Exercise 5.51 on page 5-30: The second is invalid. We may see the following counterexample:



It's easy to check that $\Diamond \to \Diamond q$ is true in world w but $\Box(p \to q)$ is not. The first implication is valid. Suppose in an arbitrary world w of a model, $K(p \to q)$ and $\Diamond p$ are both true. Then $p \to q$ is true in every world u which is accessible from w. And there exists a world v which is accessible from w such that p is true in v. But $p \to q$ must be true in v. Then, by Modus Ponens, we have q is true in v as well. This shows $\Diamond q$ is true in w, as required. The formal proof in logic K is easy since we have proved Exercise 5.46. Then $\Box(p \to q) \wedge \Diamond p \to q$ is just an instance of the proved theorem which is equivalent to $\Box(p \to q) \to (\Diamond p \to \Diamond q)$.

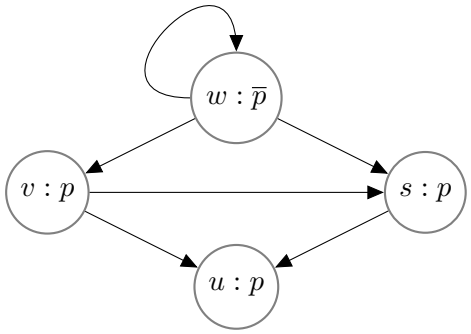
Exercise 5.52 on page 5-30:

(1) $\vdash \Box(\neg p \to q) \to (\Box\neg p \to \Box q)$ distribution axiom

(2) $\vdash \Box(p \vee q) \to (\neg\Box\neg p \vee \Box q)$ prop logic, 1

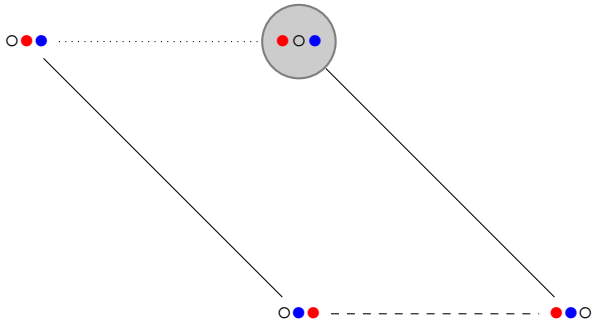
(3) $\vdash \Box(p \vee q) \to (\Diamond p \vee \Box q)$ definition of $\Diamond$ from , 2

Exercise 5.56 on page 5-33: Please see the following example model:



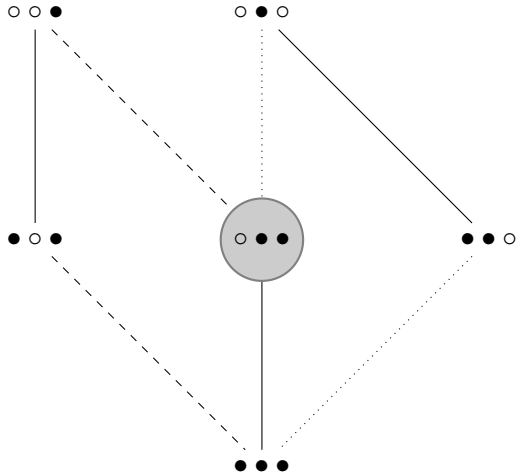
If this model is called M, then $M|\Diamond p$ will be a submodel of M with world u (and relevant relations) deleted, $(M|\Diamond p)|\Diamond p$ will exclude world s as well. And after updating with $\Diamond p$ three times $(((M|\Diamond p)|\Diamond p)|\Diamond p)$, there will be only one world w with its reflexive arrow left.

Exercise 5.58 on page 5-35: If player 2 is not treated as informative, then it means that she may have the blue card. Nothing can be updated upon the original model just after player 2's question. But then, after player 1's truthful answering 'No', all the states where player 1 has the blue card will be eliminated. We may see the following updated model:

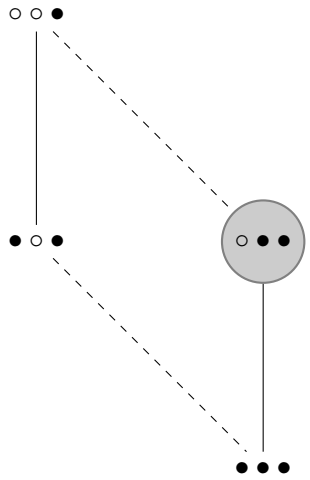


In the actual state of this new model, only player 2 knows what the card distribution is. Player 1 is still unclear about who owns the blue card. Player 3 cannot distinguish the actual state from the one where player 2 has the red card.

Exercise 5.59 on page 5-35: For the first round, 1 answers "I don't know" and then 2 says the same answer. But child 3 knows that she is dirty after hearing kid 2's answer, so she says "I know". Next, for the second round, child 1 says "I still don't know" and then child 2 answer "I don't know either". The original model and updated model after father's announcement is the same as in the original puzzle. The following diagram is the updated model after child 1 says "I don't know":

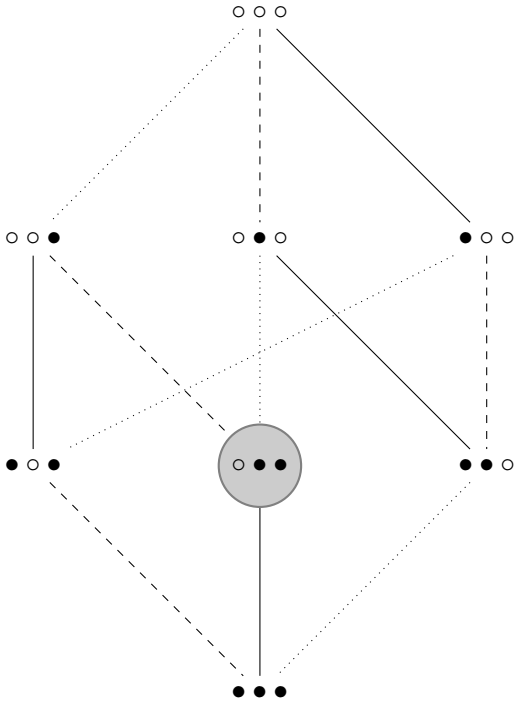


After 2's answer "I don't know", the above model is becoming smaller as

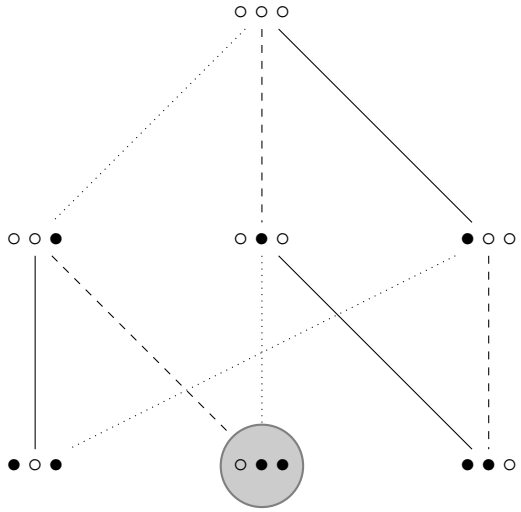


And then child 3 says “I know”, but this will not change the current model. The further announcements of child 1 and 2 in turn will not update the model either. It means that child 1 and 2 cannot figure out whether they are dirty or not.

Exercise 5.60 on page 5-35: Child 1 sees two dirty children. With their father’s public announcement, she can conclude that she is clean, so she answers “yes” in the first round. But the two dirty children do not know in the first round whether they are clean or not because each of them sees a clean and a dirty face of other two children. However, after child 1’s answer, the two dirty children know that they are not clean. For each of them would think: if I were clean, the first child couldn’t know that she was clean, so I must be dirty. Please see the following update sequence:



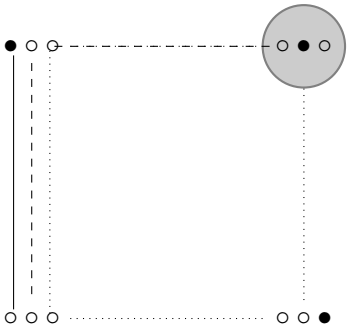
After father's announcement "At least one of you is clean", the initial model becomes as



Then child 1 says that she knows. The above model is further updated to the following final model with only one actual state:

Then child 2 and 3 know they are dirty, but these announcements have no further update effect on the above model.

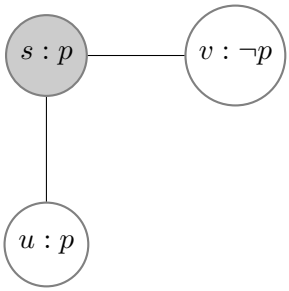
Exercise 5.61 on page 5-36: The top person's announcement is true, actually his color is white. After top person's announcement, the middle person knows his color of hat (red) as well. Then, the bottom person knows that his color is different from that of middle person although he still does not know what the actual color is. Please see the update diagrams:



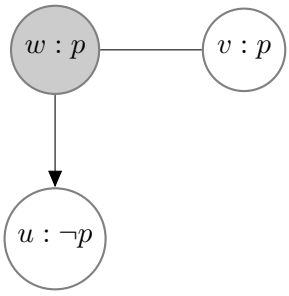
Here the real, dashed and dotted lines represent epistemic relations of top, middle and bottom men respectively. After the announcement from the person at the top that he knows the color of his hat, the original model becomes as

And then the middle person says that he knows the color of his hat. If he tells the bottom person what his actual color is, then the bottom person can also know the color of his hat.

Exercise 5.66 on page 5-38: Find a pointed model (M, s) where $M|\neg\Box p \models \neg\Box p$, and a pointed model (N, t) where $N|\neg\Box p \models \Box p$. We may see the following model M:



The real lines represent equivalence relation. It's easy to see that after the update with $\neg\Box p$, $\neg\Box p$ will be valid in the new model. For the second case, we may have the following model N:



There is only one direction arrow from world t to world u and no reflexive arrow in u. We know after the update with $\neg\Box p$, u and v will be eliminated since $\Box p$ is true in those worlds of the original model. So $\Box p$ is valid in the updated new model.

Exercise 5.74 on page 5-42:

(1) $\langle !\varphi\rangle p\& \leftrightarrow \& \varphi \wedge p$

(2) $\langle !\varphi\rangle\neg\psi\& \leftrightarrow \& \varphi \wedge \neg\langle !\varphi\rangle\psi$

(3) $\langle !\varphi\rangle(\psi \vee \chi)\& \leftrightarrow \& \langle !\varphi\rangle\psi \vee \langle !\varphi\rangle\chi$

(4) $\langle !\varphi\rangle\Diamond_i\psi\& \leftrightarrow \& \varphi \wedge \Diamond_i(\varphi \wedge \langle !\varphi\rangle\psi)$

Solutions to Exercises from Chapter 6

Exercise 6.1 on page 6-10: $b^{\smile}; a^{\smile}$.

Exercise 6.5 on page 6-14: maternal grandfather.

Exercise 6.7 on page 6-14: $\supseteq$.

Exercise 6.8 on page 6-14: $R_1 \circ R_2 = \{(s, s') \mid$ there is some $s_0 \in S : (s, s_0) \in R_1$and $(s_0, s') \in R_2\}$. $(R_1 \circ R_2)^{\smile} = \{(s', s) \mid$ there is some $s_0 \in S : (s, s_0) \in R_1$and $(s_0, s') \in R_2\} = {R_2}^{\smile} \circ {R_1}^{\smile}$.

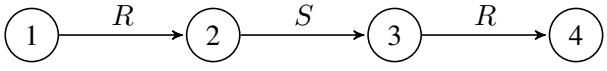
Exercise 6.10 on page 6-15: $R_1 = \{(1, 2), (3, 4)\}; R_2 = \{(1, 3)\}; R_3 = \{(3, 1)\}$

Exercise 6.11 on page 6-15: $R_1 \circ (R_2 \cup R_3) = \{(s, s') \mid$ there is some $s_0 \in S : (s, s_0) \in R_1$and $(s_0, s') \in R_2 \cup R_3\}$. $(s_0, s') \in R_2 \cup R_3$ means that $(s_0, s') \in R_2$ or $(s_0, s') \in R_3$. This means $R_1 \circ R_2$ or $R_1 \circ R_3$, that is exactly $(R_1 \circ R_2) \cup (R_1 \circ R_3)$.

Exercise 6.12 on page 6-15: $R^{\smile\smile} = \{(x, y) \in S^2 \mid (y, x) \in R^{\smile}\} = \{(x, y) \in S^2 \mid (x, y) \in R\} = R$

Exercise 6.13 on page 6-15: $(R_1 \cup R_2)^{\smile} = \{(x, y) \in S^2 \mid (y, x) \in (R_1 \cup R_2)\} = \{(x, y) \in S^2 \mid (y, x) \in R_1$ or $(y, x) \in \cup R_2)\} = {R_1}^{\smile} \cup {R_2}^{\smile}$.

Exercise 6.15 on page 6-16: The relation $(R \cup S)^*$ allow you to choose between R and S any number of times so, for example, $R \circ S \circ R$ is allowed. The relation $R^* \circ S^*$ tells you to apply R any number of times, *and then* apply S any number of times, so $R \circ S \circ R$ is not allowed. The following model is a counter-example:



We have $(1, 4) \in (R \cup S)^*$, but $(1, 4) \notin R^* \circ S^*$.

Exercise 6.16 on page 6-17: $R = \{(1, 3)\}; S = \{(3, 1), (1, 2)\}$. We can see that $(1, 2) \in R^* \circ S^* but \notin (R \circ S)^*$.

Exercise 6.17 on page 6-17: loop 'repeat a until φ' is interpreted as:

$$(R_{?\neg\varphi} \circ R_a \circ R_a)^* \circ R_{?\varphi}.$$

Exercise 6.19 on page 6-19: $\alpha^0 :=?\top$ and $\alpha^n := \alpha^{n-1}; \alpha$, for any $n > 0$.

Exercise 6.21 on page 6-21: For the state 0 of both models,

(1) $\langle a; d\rangle\surd$. True in the second model

(2) $[a; d]\surd$. True in both models

(3) $[a](\langle b\rangle\top \wedge \langle c\rangle\top)$. True in both models

(4) $[a]\langle d\rangle\surd$. True in the second model

Exercise 6.22 on page 6-22: $[a]\langle b\rangle\surd$. This formula is true in state 0 of the left graph, but false in state 0 of the right one.

Exercise 6.23 on page 6-23: The answer is 'no'. First it is clear that state 1 in the left graph satisfy the same set of PDL formulas as state 2 and state 3 in the right graph since they satisfy the same atomic sentences and there is no action relations from each of them.

Next we can verify that the root state in the left graph satisfies the same formulas as the state 1 in the right graph, since they satisfy the same Boolean formulas and have the same action relation R_b with the terminated states 1 and 2 respectively after possible execution of b. But we know state 1 in the left graph satisfies the same PDL formulas as state 2 in the right graph. This guarantees that any PDL formula in the form $\langle b\rangle\varphi$ has the same truth values in state 0 of the left and state 1 in the right.

Now we can show the root state in the left satisfies the same set of PDL formulas as the root state in the right. It's clear to check Boolean cases. The crucial cases are action modal formulas. As for the form of $\langle a\rangle\varphi$, suppose it is true in the root of the left. Then φ must be true in the same root as well. It can be guaranteed that φ is also true in state 1 of the right by the result we have just verified. So $\langle a\rangle\varphi$ is satisfied in the root of the right. As for the form of $\langle b\rangle\varphi$, suppose it is true in the root of the left. Then φ must be true in state 1 of the same graph. But we have just showed state 1 in the left satisfies the same formulas of state 3 of the right. This guarantees φ is also true in 3 of the right. So we get $\langle b\rangle\varphi$ is satisfied in the root of the right. For the converse, we can similarly verify that every true action modal formula in the root of the right is also satisfied in the root of the left.

Exercise 6.27 on page 6-24: $M \models_w \langle a; b\rangle\top$ is equivalent to $w \in [\![\langle a; b\rangle\top]\!]^{\mathcal{M}}$. From the semantic definition we have: $[\![\langle a; b\rangle\top]\!]^{\mathcal{M}} = \{w \in W_{\mathcal{M}} \mid \exists v \in W_{\mathcal{M}} : (w, v) \in [\![(a; b)]\!]^{\mathcal{M}}$ and $v \in [\![\top]\!]^{\mathcal{M}}\} = \{w \in W_{\mathcal{M}} \mid \exists v \in W_{\mathcal{M}} : (w, v) \in [\![a]\!]^{\mathcal{M}} \circ [\![b]\!]^{\mathcal{M}}\} = \{w \in W_{\mathcal{M}} \mid \exists v \in W_{\mathcal{M}} : (w, v) \in \{(x, y) \in W^2_{\mathcal{M}} \mid \exists z \in W_{\mathcal{M}}((x, z) \in \stackrel{a}{\rightarrow}_{\mathcal{M}} \wedge (z, y) \in \stackrel{b}{\rightarrow}_{\mathcal{M}})\}\} = \{w \in W_{\mathcal{M}} \mid \exists v \in W_{\mathcal{M}} : (w, v) \in \{(x, y) \in W^2_{\mathcal{M}} \mid \exists z \in W_{\mathcal{M}}((x, z) \in \stackrel{a}{\rightarrow}_{\mathcal{M}} \wedge (z, y) \in \stackrel{b}{\rightarrow}_{\mathcal{M}})\}\} = \{w \in W_{\mathcal{M}} \mid$

$\exists v \in W_{\mathcal{M}} : (w,v) \in \stackrel{a}{\rightarrow}_{\mathcal{M}}$ and $v \in \{u \in W_{\mathcal{M}} \mid (v,u) \stackrel{b}{\rightarrow}_{\mathcal{M}}$ and $u \in W_{\mathcal{M}}\}\} = \{w \in W_{\mathcal{M}} \mid \exists v \in W_{\mathcal{M}} : (w,v) \in \stackrel{a}{\rightarrow}_{\mathcal{M}}$ and $v \in [\![\langle b\rangle\top]\!]^{\mathcal{M}}\} = [\![\langle a\rangle\langle b\rangle\top]\!]^{\mathcal{M}}$. $[\![\langle a\rangle\langle b\rangle\top]\!]^{\mathcal{M}}$ is equivalent to $M \models_w \langle a\rangle\langle b\rangle\top$.

Exercise 6.28 on page 6-24:

(1) $[\![?p]\!] = \{(1,1),(2,2)\}$

(2) $[\![?(p \vee q)]\!] = \{(1,1),(2,2),(4,4)\}$

(3) $[\![a;b]\!] = \{(1,4)\}$

(4) $[\![b;a]\!] = \{(1,4)\}$

Exercise 6.29 on page 6-25:

(1) List the states where the following formulas are true:

   a. $\neg p$ is true in state 1 and 3

   b. $\langle b\rangle q$ is true in state 2

   c. $[a](p \rightarrow \langle b\rangle q)$ is true in state 2, 3 and 4

(2) Give a formula that is true only at state 4.

$$(p \wedge \neg q) \wedge [b]\bot$$

(3) Give all the elements of the relations defined by the following action expressions:

   a. $b;b$: $\{(2,4)\}$

   b. $a \cup b$: $\{(1,2),(1,4),(2,2),(4,4),(2,3),(3,4)\}$

   c. a^*: $\{(1,1),(2,2),(3,3),(4,4),(1,2),(1,4)\}$

(4) Give a PDL action expression that defines the relation $\{(1,3)\}$ in the graph.

$$?(\neg p \vee \neg q); a; ?p; b$$

Exercise 6.30 on page 6-25: $[\![\beta\check{}; \alpha\check{}]\!]^{\mathcal{M}} = [\![\beta\check{}]\!]^{\mathcal{M}} \circ [\![\alpha\check{}]\!]^{\mathcal{M}} = \{(s,t) \mid \exists u \in S_{\mathcal{M}}((s,u) \in [\![\beta\check{}]\!]^{\mathcal{M}} \wedge (u,t) \in [\![\alpha\check{}]\!]^{\mathcal{M}}\} = \{(s,t) \mid \exists u \in S_{\mathcal{M}}((u,s) \in [\![\beta]\!]^{\mathcal{M}} \wedge (t,u) \in [\![\alpha]\!]^{\mathcal{M}}\} = \{(s,t) \mid (t,s) \in [\![\alpha]\!]^{\mathcal{M}} \circ [\![\beta]\!]^{\mathcal{M}}\} = \{(s,t) \mid (t,s) \in [\![\alpha;\beta]\!]^{\mathcal{M}}\} = [\![(\alpha;\beta)\check{}]\!]^{\mathcal{M}}$

$[\![\beta\check{} \cup \alpha\check{}]\!]^{\mathcal{M}} = [\![\beta\check{}]\!]^{\mathcal{M}} \cup [\![\alpha\check{}]\!]^{\mathcal{M}} = [\![\beta\check{}]\!]^{\mathcal{M}} \cup [\![\alpha\check{}]\!]^{\mathcal{M}} = \{(s,t) \mid (t,s) \in [\![\alpha]\!]^{\mathcal{M}}\} \cup \{(s,t) \mid (t,s) \in [\![\beta]\!]^{\mathcal{M}}\} = \{(s,t) \mid (t,s) \in [\![\beta]\!]^{\mathcal{M}} \cup [\![\alpha]\!]^{\mathcal{M}}\} = [\![(\beta \cup \alpha)\check{}]\!]^{\mathcal{M}}$

$[\![(\alpha^*)\check{}]\!]^{\mathcal{M}} = \{(s,t) \mid (t,s) \in [\![\alpha^*]\!]^{\mathcal{M}}\} = \{(s,t) \mid (t,s) \in ([\![\alpha]\!]^{\mathcal{M}})^*\} = \{(s,t) \mid (t,s) \in \bigcup_{n\in\mathbb{N}}([\![\alpha]\!]^{\mathcal{M}})^n\} = \bigcup_{n\in\mathbb{N}}([\![\alpha\check{}]\!]^{\mathcal{M}})^n = ([\![\alpha\check{}]\!]^{\mathcal{M}})^*$

Exercise 6.31 on page 6-25: $\alpha^{\smile} = \begin{cases} \beta^{\smile} \cup \gamma^{\smile} & \text{if } \alpha = (\beta \cup \gamma)^{\smile} \\ \beta^{\smile}; \gamma^{\smile} & \text{if } \alpha = (\beta; \gamma)^{\smile} \\ ?\varphi & \text{if } \alpha = ?\varphi^{\smile} \\ (\beta^{\smile})^* & \text{if } \alpha = (\beta^*)^{\smile} \\ \beta & \text{if } \alpha = \beta^{\smile\smile} \\ \{(t,s) \mid (s,t) \in a\} & \text{if } \alpha = a \end{cases}$

Exercise 6.37 on page 6-29: $\vdash \langle \alpha^* \rangle \varphi \leftrightarrow \varphi \vee \langle \alpha \rangle \langle \alpha^* \rangle \varphi$.

Exercise 6.39 on page 6-30: Let w be an arbitrary state in an arbitrary LTS. Assume φ is true in w. If w is an a-isolated state then the consequent is true. If not then take an arbitrary v such that $(w, v) \in a$ then $(v, w) \in a^{\smile}$ hence $\langle a^{\smile} \rangle \varphi$ must be true at v. As v was arbitrary $[a]\langle a^{\smile} \rangle \varphi$ must be true at w. Let w be an arbitrary state in an arbitrary LTS. Assume φ is true in w. If w is an a-isolated state then the consequent is true. If not, then take an arbitrary v such that $(w, v) \in a^{\smile}$ then $(v, w) \in a$ hence $\langle a \rangle \varphi$ must be true at v. As v was arbitrary $[a^{\smile}]\langle a \rangle \varphi$ must be true at w.

Exercise 6.40 on page 6-32: $[\![i]\!]_s = i, [\![v]\!]_s = s(v), [\![a_1 + a_2]\!]_s = [\![a_1]\!]_s + [\![a_2]\!]_s, [\![a_1 * a_2]\!]_s = [\![a_1]\!]_s * [\![a_2]\!]_s, [\![a_1 - a_2]\!]_s = [\![a_1]\!]_s - [\![a_2]\!]_s$.

Exercise 6.41 on page 6-33: There are three possible procedure stages: (1) both drawn pebbles are black (2) both drawn pebbles are white (3) there have been drawn a white and a black pebble. For (1) the number of white pebbles remains unchanged, for (2) the number of white pebbles remains odd ($n' = n - 2$), for (3) the number of white pebbels remains unchanged ($n' = n - 1 + 1$). Hence the "oddness" property of the number of white pebbels is invariant during any number of executions of the drawing procedure. Therefore, if there is only one pebble left it must be white.

Exercise 6.46 on page 6-38: Actually we can prove that the root 0 of the left graph is bisimilar with the the root 0 in the right. Let $Z = \{(0,0), (0,1), (1,2), (1,3)\}$ between states of two models. First these two root states satisfy the same atomic sentences. With the help of the detailed solution in Exercise 6.23, the zigzag relation Z completes the square and satisfies the definition of bisimulation between those two models.

Exercise 6.47 on page 6-39: We show it by induction on the structure of φ. Let s and t be states in the two models M and N respectively. For the base case, it is clear that s and t satisfy the same atomic sentences since these two states are bisimilar. Proofs for Boolean cases are routine. We only consider crucial case (the modal case, i.e., $\varphi = \langle a \rangle \psi$) here. For one direction, suppose that $\langle a \rangle \psi$ is true in s. Then there is a state s_1 satisfying $s \xrightarrow{a} s_1 \in R_M$ in the one model and ψ is true in state s_1. By the forth (Zig) condition of 'bisimulation', we get that there exist a zigzag relation Z between two models and a state t_1 in the other model, satisfying $t \xrightarrow{a} t_1 \in R_N$ and $s_1 Z t_1$. Then psi is also true in state t_1 by induction hypothesis. This means φ is satisfied in state t, as required. For the other direction, suppose that $\langle a \rangle \psi$ is true in t. Then there is a state t' satisfying $t \xrightarrow{a} t' \in R_N$ in the other model and ψ is true in state t'. By the back (Zag) condition of 'bisimulation', we have that there exist a state s' in the one model such that $s \xrightarrow{a} s' \in R_M$ and $s' Z t'$. Similarly by the induction hypothesis we can get psi is true in state s'. This means φ is true as well in state s, as required.

Exercise 6.48 on page 6-39: We show it by using induction on the structure of α. For the base case, that α is an atomic program, such as a. Let s, t be states of model M and N respectively with sCt (C is a bisimulation relation between M and N). It's clear to see that a is safe for bisimulation: just let $\alpha_M = \alpha_N = \xrightarrow{a}$. If α has the form of $\beta;\gamma$. Then $\alpha_M = \beta_M \circ \gamma_M$, and $\alpha_N = \beta_N \circ \gamma_N$. Next we show $\beta;\gamma$ satisfies the Zig condition of safety. Suppose $s\alpha_M s'$ for some $s' \in S_M$. We have $s\beta_M \circ \gamma_M s'$, implying that there exists an state $w \in S_M$ satisfying $s\beta_M w$ and $w\gamma_M s'$. By induction hypothesis, it follows that β and γ are both safe for bisimulation. This means there exist some u and t' in S_N with $t\beta_N u$, $u\gamma_N t'$ and wCu, $s'Ct'$. But it is just $t\beta_N \circ \gamma_N t'$ = $t\alpha_N t'$ with $s'Ct'$, as required. Similarly we can prove that $\beta;\gamma$ satisfies the converse (Zag) condition of safety. And proof for the case $\alpha = \beta \cup \gamma$ is also routine. Details of checking are omitted. Now we check the 'test' case, that is, $\alpha = ?\varphi$. We know that s and t satisfy the same set of formulas since they are bisimilar. It means that if $s?\varphi_M s'$ for some $s' \in S_M$(s' is just s since the program is 'test'), then there must have $t?\varphi_N t$. It is clear that sCt. It shows that the Zig condition of the safety is satisfied. It is similar and easy to have the Zag condition satisfied in 'test' case. This completes the proof.

Solutions to Exercises from Chapter 7

Exercise 7.5 on page 7-7:

Please see the proof in the solution for Exercise 4.26. Now consider the strategy for Verifier: Falsifier tries to find that every object in a finite network which cannot reach some node in at most 2 steps. It's enough for Verifier to find just one node that can reach every node in at most 2 steps. Verifier may do this inductively on the number of nodes since it is finite. For the 1 and 2 node cases, it is easy to see that there is a 'Great Communicator' since the network is reflexive and connected. Now consider the $n+1$ node case. Verifier knows that there is a 'Great Communicator' in the subgraph of every n nodes and by induction hypothesis she can pick out that 'Great node' in n node subgraph. If the 'Great node' has a link into the $(n+1)^{th}$ node, just pick out it as a 'Great Communicator' for the whole $n+1$ node network. If the 'Great node' has no link into the $(n+1)^{th}$ node, then by connectedness, the latter must have a link into the 'Great node' of the n node subgraph. Since the 'Great node' of n node subgraph can reach every other node in at most two steps and the network is transitive, the $(n+1)^{th}$ node can reach every node in n node subgraph including the 'great node' (in one step). Then Verifier should pick out the $(n+1)^{th}$ node as a 'Great Communicator' for the whole $n+1$ node network.

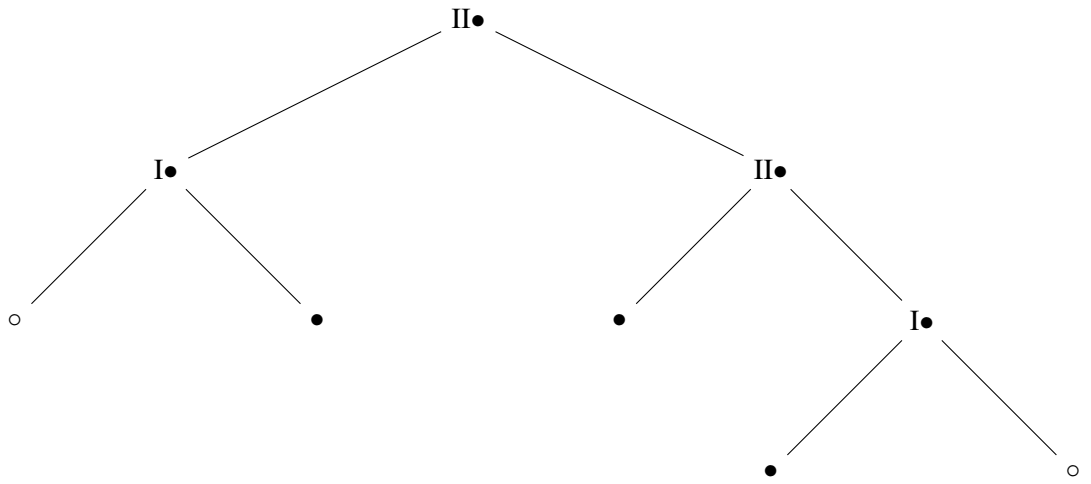
Exercise 7.7 on page 7-7:

Proof. We prove this lemma by induction on the construction of formulas. For the base case, that is, φ is an atom such as Pd. If Pd is true in $(\mathcal{M}, s)$, then Verifier has a winning strategy by the definition of evaluation games. If Pd is false in $(\mathcal{M}, s)$, then Falsifier has a winning strategy in game$(\varphi, \mathcal{M}, s)$. Boolean cases are easy to demonstrate, we just choose disjunction as an example here. If $\varphi = \psi \vee \chi$ is true in $(\mathcal{M}, s)$, then ψ or χ is true in $(\mathcal{M}, s)$. By induction hypothesis, we know that Verifier has a winning strategy in game$(\psi, \mathcal{M}, s)$ or in game$(\chi, \mathcal{M}, s)$. Now it's turn for Verifier to proceed since φ is a disjunction formula. Verifier just chooses a subgame

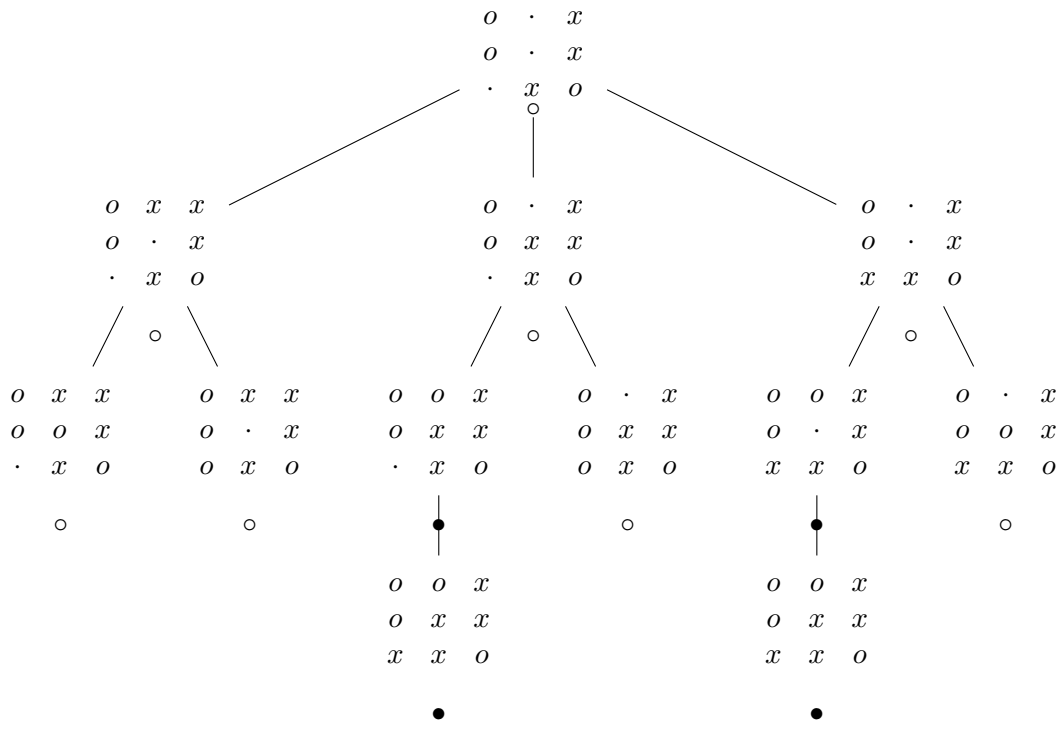
game($\psi, \mathcal{M}, s$) or game($\chi, \mathcal{M}, s$) which is winning strategy for her to continue. It shows that Verifier has a winning strategy in the game($\varphi, \mathcal{M}, s$). If $\varphi = \psi \vee \chi$ is false in $(\mathcal{M}, s)$, then ψ and χ are both false in $(\mathcal{M}, s)$. By induction hypothesis, Falsifier has a winning strategy in both game($\psi, \mathcal{M}, s$) and game($\chi, \mathcal{M}, s$). It shows that Verifier will lose in game($\varphi, \mathcal{M}, s$) since whatever she chooses to continue, Falsifier has a winning strategy in each subgame. This means Falsifier has a winning strategy in game($\varphi, \mathcal{M}, s$). Next we consider the crucial case that φ is a quantifier formula such as $\exists x\psi(x)$. If it is true in $(\mathcal{M}, s)$, then there exists an object d in the domain of $\mathcal{M}$ such that $\psi(d)$ is true in $(\mathcal{M}, s)$. By induction hypothesis, we know that Verifier has a winning strategy in game($\psi(d), \mathcal{M}, s$). Now for the game($\varphi, \mathcal{M}, s$), it's turn for Verifier to choose an object to play to continue. It's safe for her to choose the object d to guarantee winning the game since she has a winning strategy in game($\psi(d), \mathcal{M}, s$). If $\exists x\psi(x)$ is false in $(\mathcal{M}, s)$, then for every object f in the domain of $\mathcal{M}$ such that $\psi(f)$ is false in $(\mathcal{M}, s)$. By induction hypothesis, Falsifier has a winning strategy in game($\psi(f), \mathcal{M}, s$) for every f. Then for the game($\exists x\psi(x), \mathcal{M}, s$), whatever object Verifier chooses, Falsifier can always win, as required.

Exercise 7.8 on page 7-7: For the predicate logical formulas, we can define them in terms of evaluation games by induction of their constructions. For atomic formulas such as Px, players should pick some object d according to the assignment s of model $\mathcal{M}$ as a value for variable x and test the atom Pd for truth or falsity. If it is true then Verifier wins, if it is false then Falsifier wins. Disjunctions such as $\varphi \vee \psi$ can be looked as a choice with a turn by Verifier between two games, game($\varphi, \mathcal{M}, s$) and game($\psi, \mathcal{M}, s$) and then continue to play the game that Verifier chose. Conjunctions such as $\varphi \wedge \psi$ can be defined as a game of choice as well with a turn by Falsifier: she selects one game of game($\varphi, \mathcal{M}, s$) and game($\psi, \mathcal{M}, s$) to continue. For the quantifier formulas such as $\exists x\varphi(x)$, it can be looked as a "sequential composition" game with a turn by Verifier to pick some object such as f, and then they continue to play with game($\varphi(f), \mathcal{M}, s$). If it is a formula with universal quantifiers such as $\forall x\varphi(x)$, then we can think it as a "sequential composition" game with a turn by Falsifier to pick an object such as g in the domain of model $\mathcal{M}$, and then they continue to play with game($\varphi(g), \mathcal{M}, s$). Conversely, we can give an evaluation game similarly that corresponds to a logical formula but that is not a predicate-logical formula, for example, a modal formula $\Box\varphi$. It can be looked as a "sequential composition" game with a turn by Falsifier who is going to pick some object t in the domain of model $\mathcal{M}$ that is accessible from s (or we say sRt), and then continue to play game($\varphi, \mathcal{M}, t$).

Exercise 7.10 on page 7-9 All the left nodes are cloloured black, please see the following diagram:



Exercise 7.12 on page 7-10: Let player X be player I and O player II. Then we may have the following coloured diagram.



Exercise 7.13 on page 7-11: Suppose we have two players 1 and 2. Predicates of bottom states, having a winning strategy for 1 and 2 are denoted as B, Win_1 and Win_2, and binary predicates for moves of 1 and 2 are denoted as R_1 and R_2 respectively. In each bottom state of the above game tree x, we have $Win_1(x)$ or $Win_2(x)$. In each state y before the last moves, if the next move was made by player 1 and she can reach a bottom state x where she wins, then $Win_1(y)$; if

the next move was made by player 1 and she cannot reach a bottom state x where she wins, then $Win_2(y)$; if the next move was made by player 2 and she can reach a bottom state x where she wins, then $Win_2(y)$; if the next move was made by player 2 and she cannot reach a bottom state x where she wins, then $Win_1(y)$. Hence in each state before the last moves, it can be determined that from that state on, player 1 can win or player 2 can win. Similarly it can be decided in each state before the last two moves and the root of the game. We may express this in the conjunction of the following first-order formulas:

(1) $\forall x(B(x) \to Win_1(x) \vee Win_2(x))$

(2) $\forall x \exists y(R_1xy \wedge Win_1(y) \to Win_1(x)$

(3) $\forall x \forall y(R_1xy \wedge Win_2(y) \to Win_2(x)$

(4) $\forall x \exists y(R_2xy \wedge Win_2(y) \to Win_2(x)$

(5) $\forall x \forall y(R_2xy \wedge Win_1(y) \to Win_1(x)$

From those formulas we can conclude that in the root r (actually in every state) of the game, $Win_1(r) \vee Win_2(r)$ is true.

Exercise 7.14 on page 7-11: Let player II has no winning strategy at some stage s of the game and suppose for reductio that I has no strategies for achieving a set of runs from s during all of which II never has a winning strategy for the remaining game from then on. This means from s, II has a chance to reach a winning strategy state in the remaining game from then on.

Exercise 7.15 on page 7-11: The natural moves of defense and attack in the epistemic evaluation game will be indicated henceforth as

$$\text{game}(\varphi, \mathcal{M}, s)$$

The moves of evaluation games follow the inductive construction of formulas. They involve some typical actions that occur in games, such as *choice, switch*, and *continuation*, coming in dual pairs with both players **V** (Verifier) and **F** (Falsifier) allowed the initiative once:

Atomic sentences $p, q, \ldots$
V wins if the atomic sentence is true in s, **F** if it is false

Disjunction $\varphi \vee \psi$:
V chooses which disjunct to play

Conjunction $\varphi \wedge \psi$:
F chooses which conjunct to play

Negation $\neg\varphi$:
Role switch between the players, play continues with respect to φ.

Next, the knowledge operators make players look at the states which are successors (epistemically accessible from) of the current state s:

Diamond $\Diamond\varphi$:

V moves to a state t which is a successor of s, and then play continues with respect to $game(\varphi, \mathcal{M}, t)$.

Box $\Box\varphi$:

The same, but now for **F**.

The game ends at atomic sentences: Verifier wins if it is true, Falsifier wins if it is false.

Exercise 7.17 on page 7-13: Blocker has the winning strategy in this new scenario. Here is one: he first cuts the link between Haarlem and Sloterdijk. Then Runner can only go to Leiden. Blocker cuts the link between Haarlem and Leiden next. After that, there are only two possible places left for Runner to go, Sloterdijk and Amsterdam.

(1) If Runner goes to Amsterdam, Blocker cuts a link between Leiden and Sloterdijk to see the next response of Runner. There are two possible subcases. (a) If Runner goes back to Leiden, Blocker should cut the other link between Leiden and Sloterdijk. Then Runner must go to Amsterdam. Blocker cuts a link between Amsterdam and Sloterdijk to see the Runner's following choice: if Runner goes to Sloterdijk, he should cut the link between Leiden and Amsterdam. Runner must move to Amsterdam and then Blocker cuts the second link between Amsterdam and Sloterdijk. So Runner cannot go to any other places. (b) If Runner goes to Sloterdijk, Blocker cuts the second link between Leiden and Sloterdijk. Then Runner can go only to Amsterdam. Blocker cuts the link between Leiden and Amsterdam next, forcing Runner to move to Sloterdijk. But then Blocker cuts a link between Amsterdam and Sloterdijk, making Runner move to Amsterdam. Blocker cuts the second link between Amsterdam and Sloterdijk, and Runner is forced to stay in Amsterdam.

(2) If Runner goes to Sloterdijk. Blocker cuts a link between Leiden and Sloterdijk. Next if Runner goes to Leiden, Blocker just cuts the second link between Leiden and Sloterdijk, forcing him to go to Amsterdam after that. This exaclty the situation in subcase (a) of case (1). If Runner goes to Amsterdam, Blocker should cut the link between Leiden and Amsterdam. As Runner must move to Sloterdijk next, then Blocker cuts the second link between Leiden and Sloterdijk. Now Runner is forced to move between Amsterdam and Sloterdijk. Blocker should wait for Runner to reach Sloterdijk, then removes a link between Amsterdam and Sloterdijk. After Runner goes back to Amsterdam, Blocker should remove the last existing link, leaving Amsterdam isolated and making Runner stay there.

Exercise 7.18 on page 7-13: Runner has a winning strategy. Runner should go to 2 first. Next if Blocker cuts a link between 3 and 4, Runner should go to 4. In order to prevent Runner visiting 3, Blocker will cut the second link between 3 and 4, then Runner should go back to 2. Now it's too late for Blocker to prevent Runner reaching 3. And if Blocker cuts the link between 2 and 4 just after the first move of Runner, Runner should go from 2 to 3. Then it's too late for Blocker to prevent Runner visiting 4.

Exercise 7.23 on page 7-15: **S** has a winning strategy. First he picks j in $\mathcal{N}$, and then takes l in

the next round. No such pattern occurs in $\mathcal{M}$, as $\{(j,2),(l,3)\}$ is not a partial isomorphism. So **D** is bound to lose.

Exercise 7.33 on page 7-21: We know that formula $[a]\langle b\rangle\surd$ is true in state 0 of the left graph, but false in state 0 of the right one. So **S** can have the following winning strategy: he first chooses 0 and then 1 in the left graph. **D** should response to choose 0 and 2 in the right graph, but $\{(0,0),(1,2)\}$ is not a bisimulation relation since 0 has an R_b successor 1 in the left model but 0 has no R_b successors in the right model.

Exercise 7.43 on page 7-28: We know, in the last round, there has two possible combinations for voting: A versus B or B versus C, and the respective results are that B wins or C wins. Then consider the first round vote, it is impossible for A (which 1 likes best) to win at last. If A wins in the first round then B will win at last. 1 knows all of the above, so it will not vote for A since he will lose at the end and it's rational for party 1 to vote for C which it prefers to outcome B. As for the party 2, it will try to make C win in the first round since C is its best choice and C can win at last if he wins in the first round. As for the party 3, obvious it prefers B to win in the last round, but C is a threat. So it will vote for A in the first round. Now we can conclude the result of this voting: C wins in the first and last round.

Exercise 7.49 on page 7-32: The formula of PDL is

$$[\sigma^*](end \to \varphi) \to [move_i]\langle\sigma^*\rangle(end \wedge \varphi)$$

Exercise 7.61 on page 7-38: Suppose that player I plays a particular action such as choosing to show the head with probability $p > \frac{1}{2}$, then the probability of his playing to show the tail $q < \frac{1}{2}$. Player II can exploit this by playing an action to just show the tail always. So the probability of gain for player II is larger than $\frac{1}{2}$ but the probability of lost is smaller than $\frac{1}{2}$. It is obvious that player II can benefit from this. If player I plays an action to show the tail with probability $p > \frac{1}{2}$, player II can just always play an action to show the head. For the same reason, he has advantages to benefit from the game. Similarly if player II plays a particular action with probability $p > \frac{1}{2}$, player I can also exploit this by playing to show always head or always tail. This shows that the resulting pair of strategies is not Nash.

Exercise 7.68 on page 7-41: For the pure strategy Nash equilibria, pairs of (H,d) and (D,h) with utilities $(3,1)$ and $(1,3)$ are two Nash equilibria for the game Hawk versus Dove. In (H,d), for the player who plays Hawk, if he switches to play Dove, the outcome will be $(2,2)$. It's smaller than the original 3 he gets. So it's not rational for Hawk to switch the current role. As for the player who plays Dove, if he switches into Hawk, the resulting outcome will be $(0,0)$. It is obvious not good for him compared with the original 1. So Dove has no motivation to switch the role either. Similarly the situation (D,h) can be analyzed.

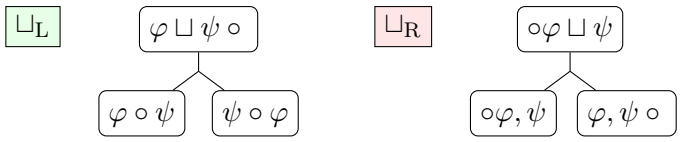
Exercise 7.76 on page 7-47: The situation can be represented by the following strategic game:

	$!\neg p$	$!\top$
$!\neg q$	$2, 2$	$3, 0$
$!\top$	$0, 3$	$1, 1$

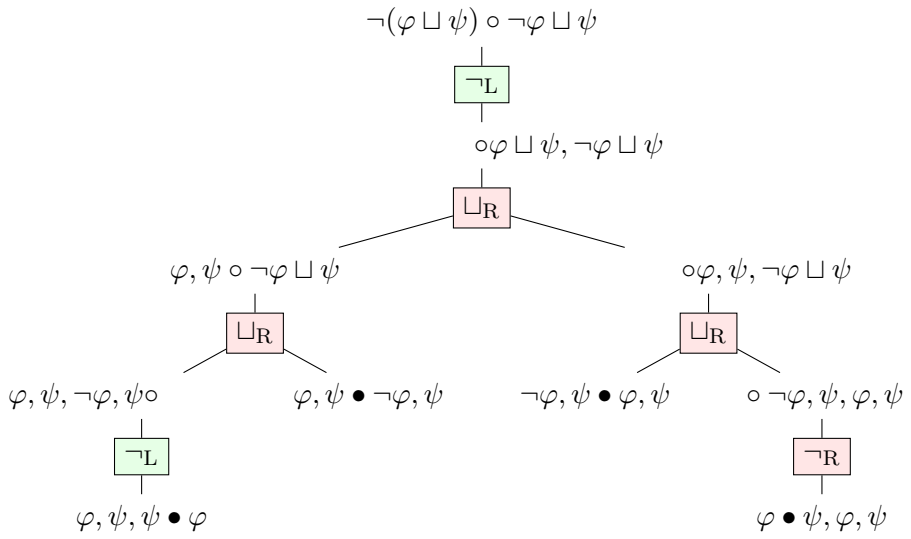
For player II (actually who knows q), if he receives information that $\neg q$ from player I, it is not useful him since he has already know that q. But if he keeps silent in the case of player I saying $\neg q$, then it is equivalent to telling player I that he knows q. A better choice for player II is to tell player I that $\neg p$. The strategy for player I is similar when he receives information that $\neg p$ from player II. So the only equilibrium there is $(!\neg p, !\neg q)$.

Solutions to Exercises from Chapter 8

Exercise 8.1 on page 8-8:

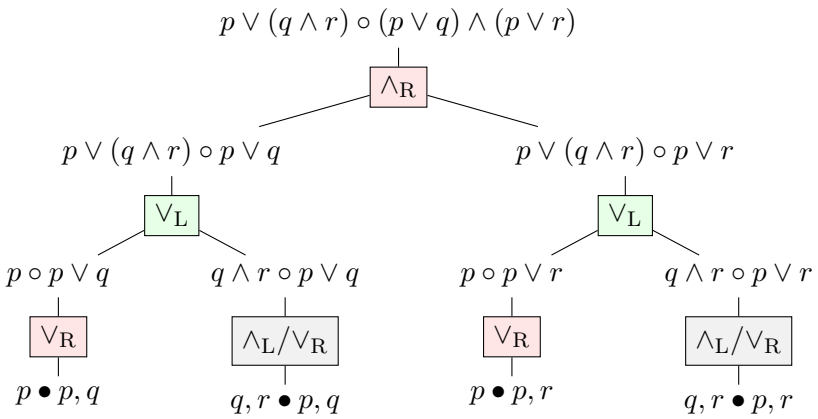


Exercise 8.2 on page 8-8: Here you have the left-to-right direction:



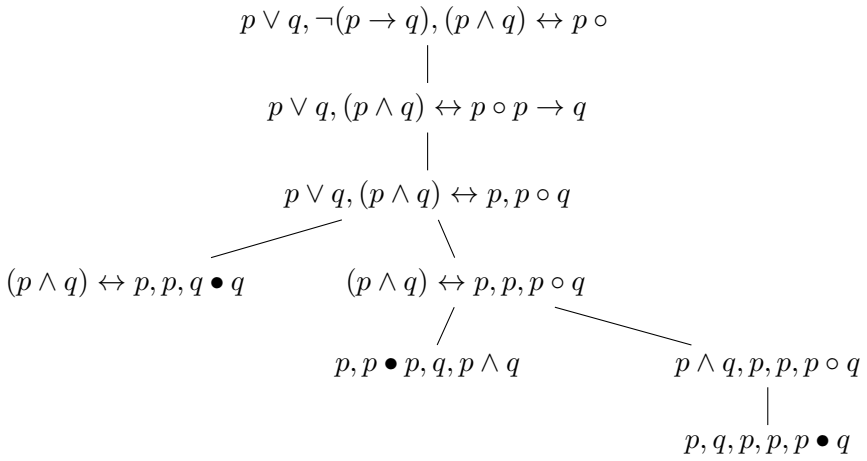
You can check that the right-to-left direction also holds.

Exercise 8.3 on page 8-9: Here you have the table for point (1):



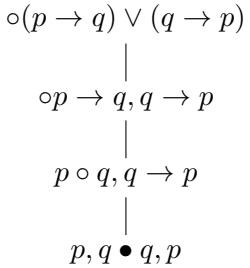
You can check that point (2) also tests positively for validity.

Exercise 8.4 on page 8-9: Here you have the table for point (2), now with implicit rules:



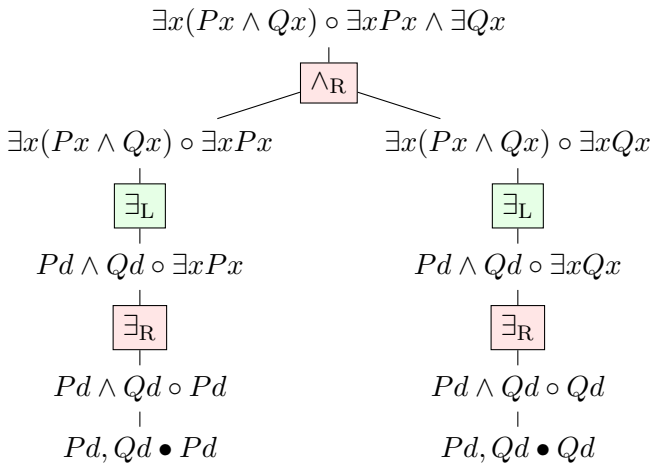
You can check that point (1) is not satisfiable either.

Exercise 8.5 on page 8-9: Here we may first check that (1) is a tautology.



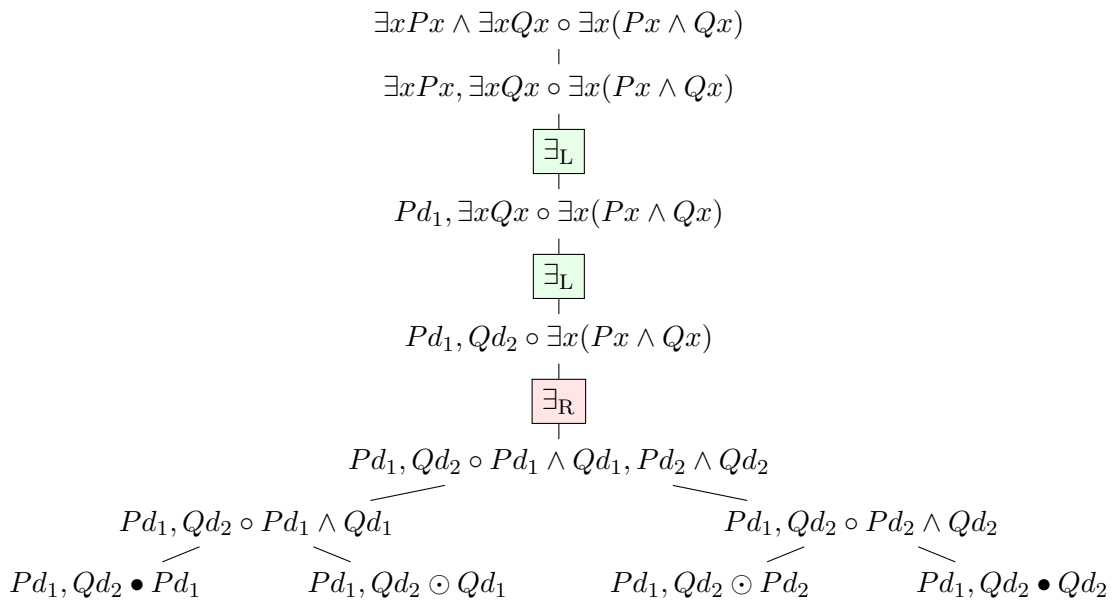
You can check that point (2) is not a tautology.

Exercise 8.6 on page 8-11: We may show it in the following table



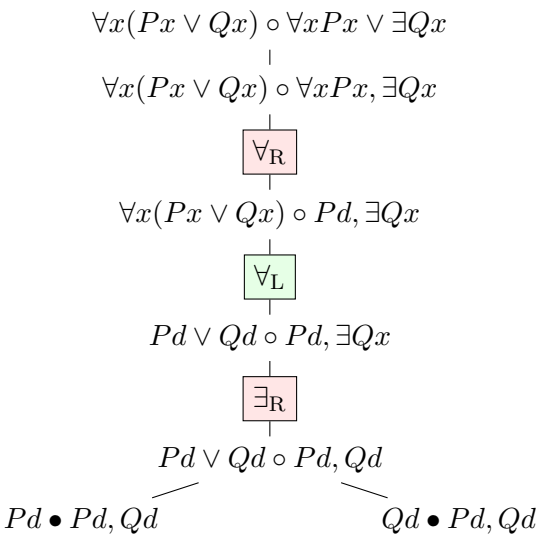
All branches in the table are closed. This means $\exists x(Px \wedge Qx) \models \exists xPx \wedge \exists Qx$.

Exercise 8.7 on page 8-12:

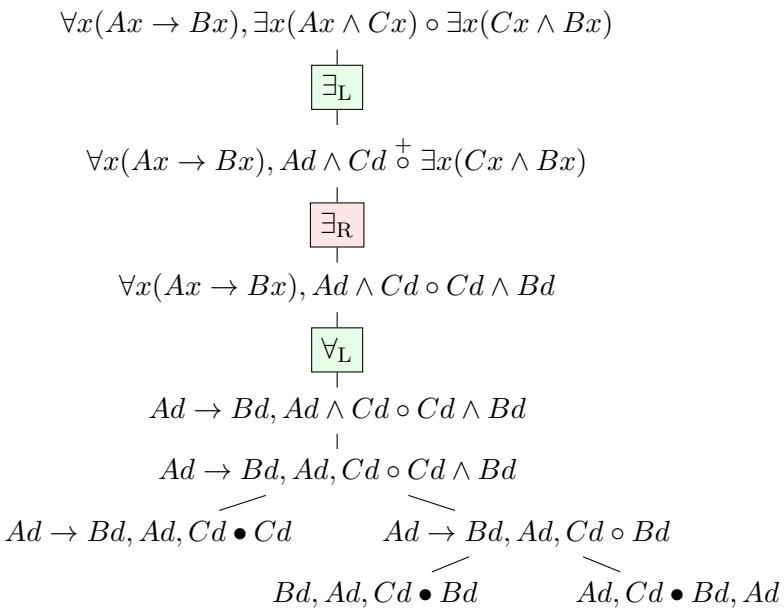


There are two open branches in the table, one of them shows that we can find two objects d_1 and d_2 in a model such that d_1 is P and d_2 is Q but d_1 is not Q.

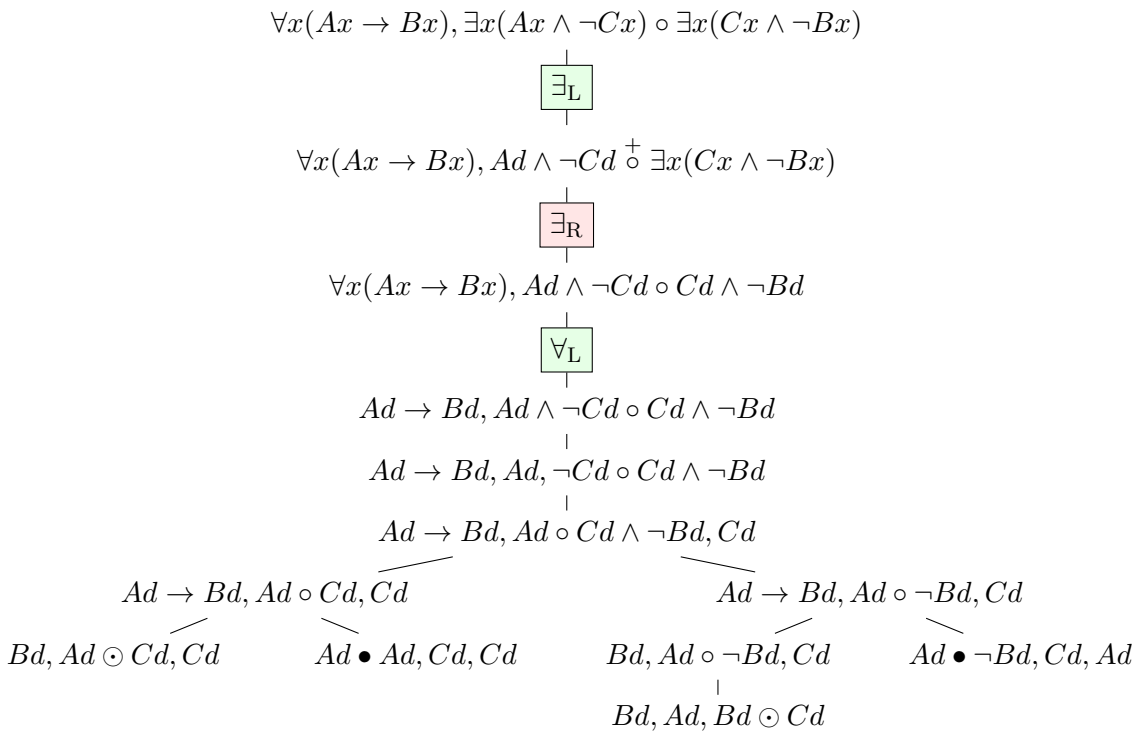
Exercise 8.8 on page 8-12:



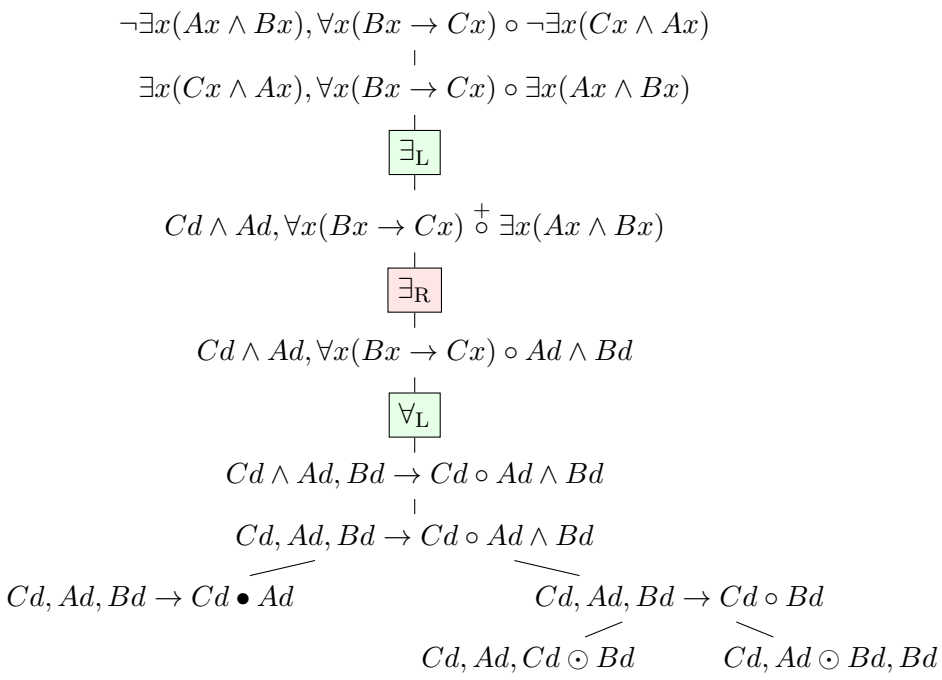
Exercise 8.9 on page 8-16: First for (1)



Next for (2):

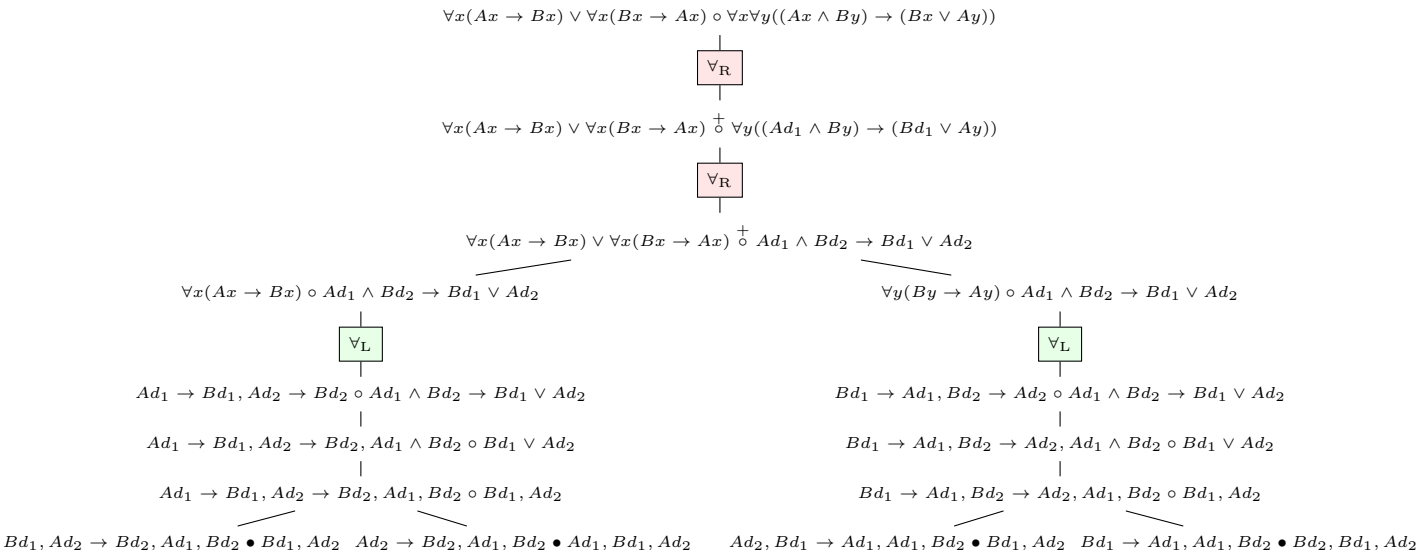


Now for (3):



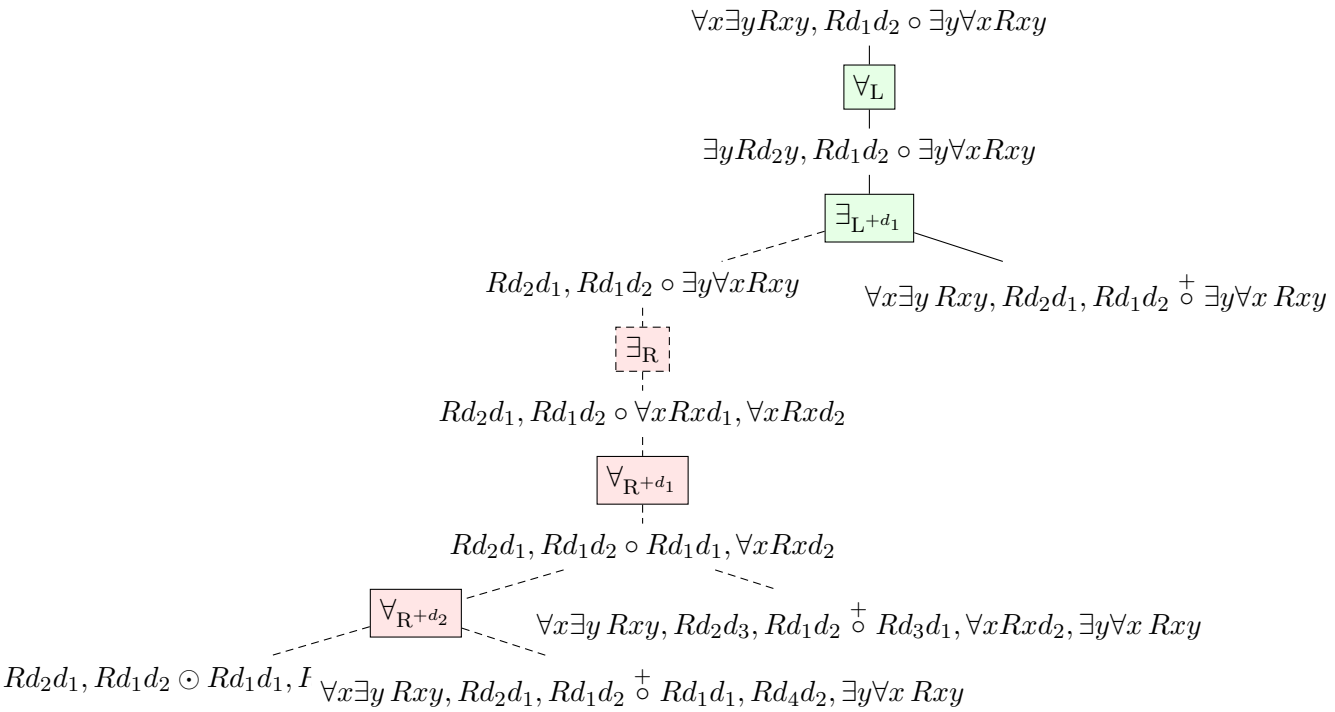
It can be seen from the above tables that (1) is valid, but (2) and (3) are not.

Exercise 8.10 on page 8-17: First please see (1) in the following table.



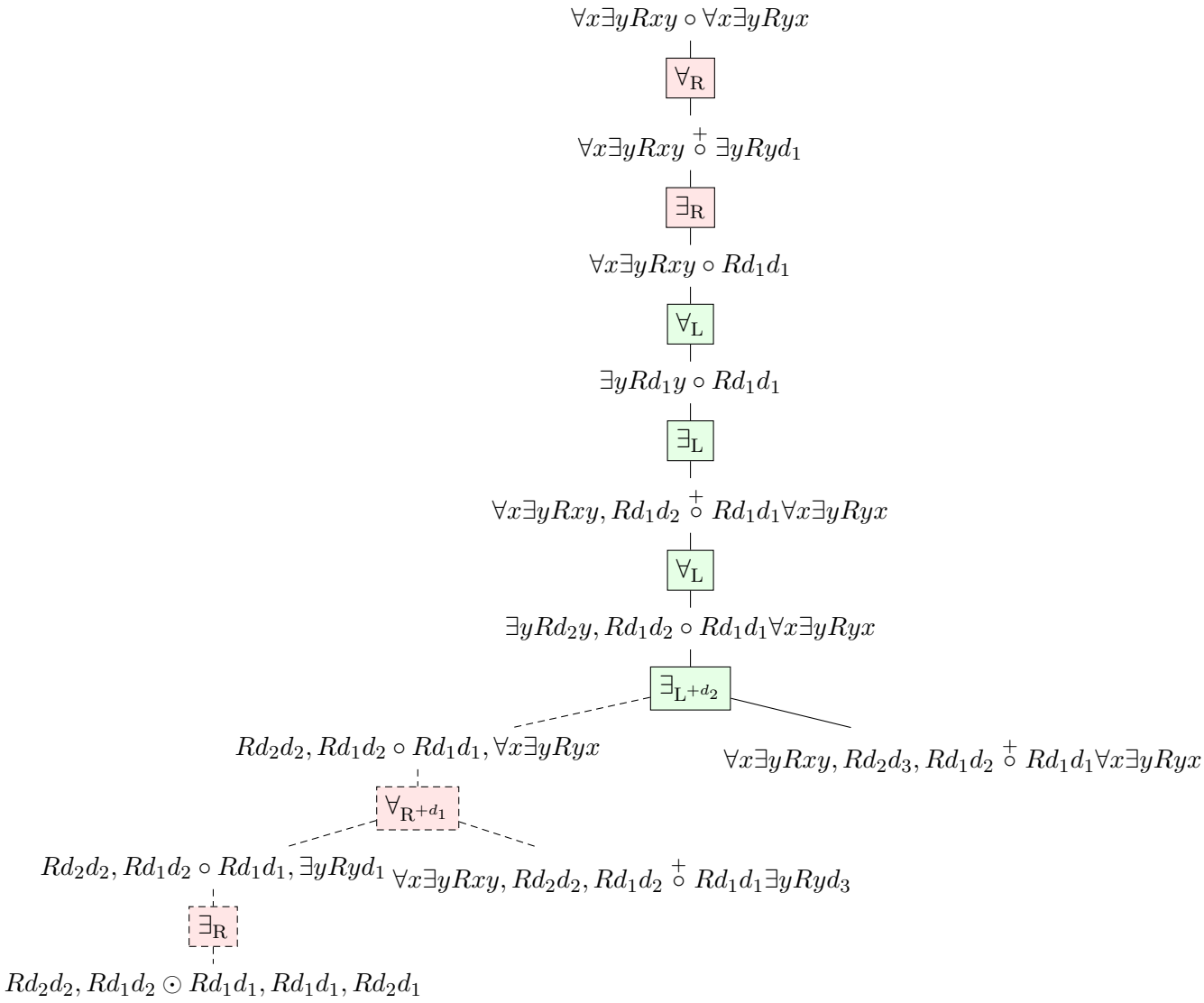
Similarly we can check that (2) also holds.

Exercise 8.11 on page 8-19: We choose for the regular branch after the second step in the text as a starting point.



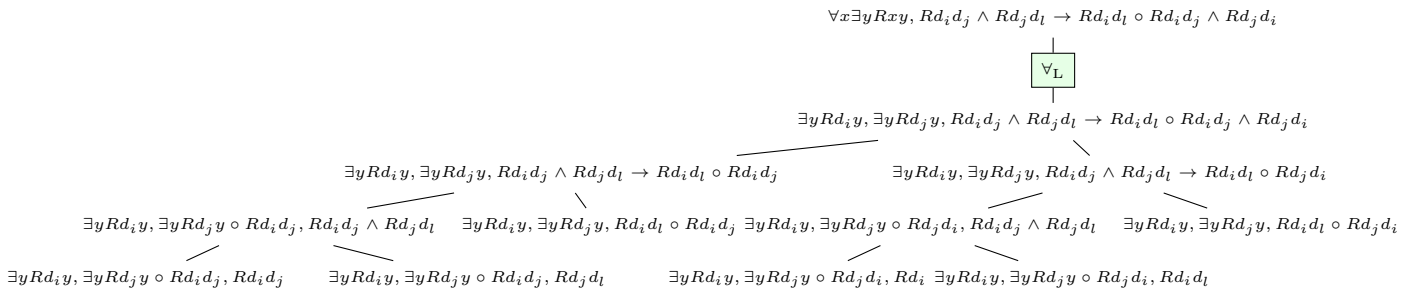
Now we get a simple counter model with two objects who are mutually related but are not related to themselves.

Exercise 8.12 on page 8-19: We only check (1) in the following table. It is similar to check (2) with try out method to get a simple counter model for it.



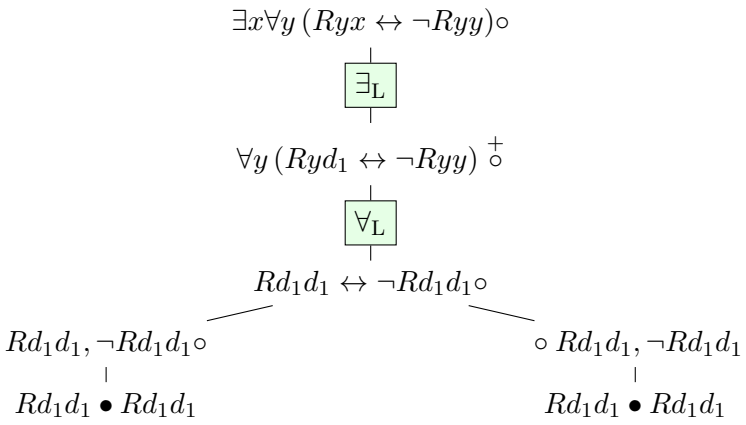
Similarly we may get a simple open branch $d_1d_2, d_1d_1 \odot d_2d_2, d_2d_1$ for $\exists x\forall y\, Rxy/\exists x\forall y\, Ryx$, showing that it is not valid.

Exercise 8.13 on page 8-19: Since the number of objects in a model is finite, suppose it is n, we can list all the objects $d_1, ..., d_n$. Then the try out method in the following tableau process must stop (for any natural number $i, j, l \leq n$).

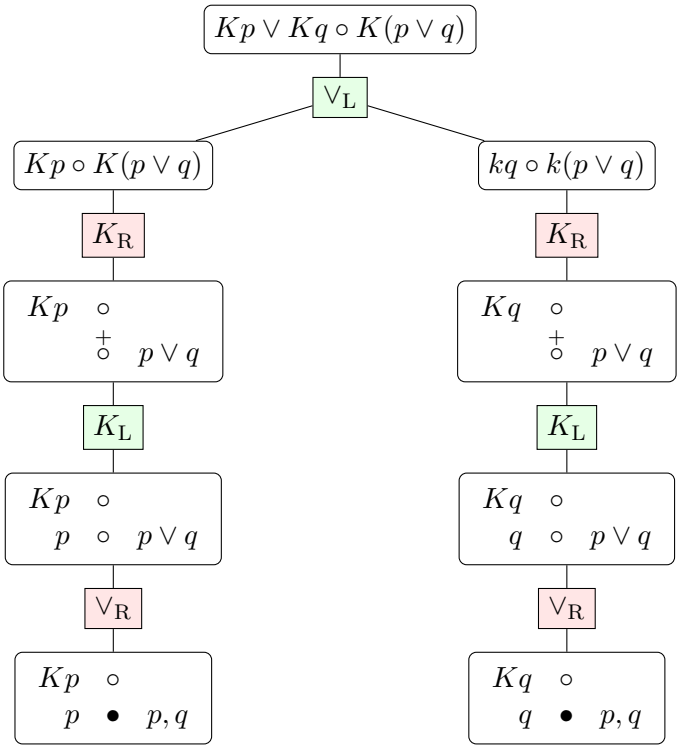


Now we only consider a branch $\exists y Rd_i y, \exists y Rd_j y \circ Rd_i d_j, Rd_j d_l$ in the above without loss of generality. For any number $k \leq n$ y get, there is an counterpart $Rd_i d_k$ in the right side since j is an arbitrary number that is less than or equal to n. Hence this branch of the above tableau is closed. Similarly we can also decide that any other branches in the tableau are closed, validating the original conclusion.

Exercise 8.14 on page 8-21:

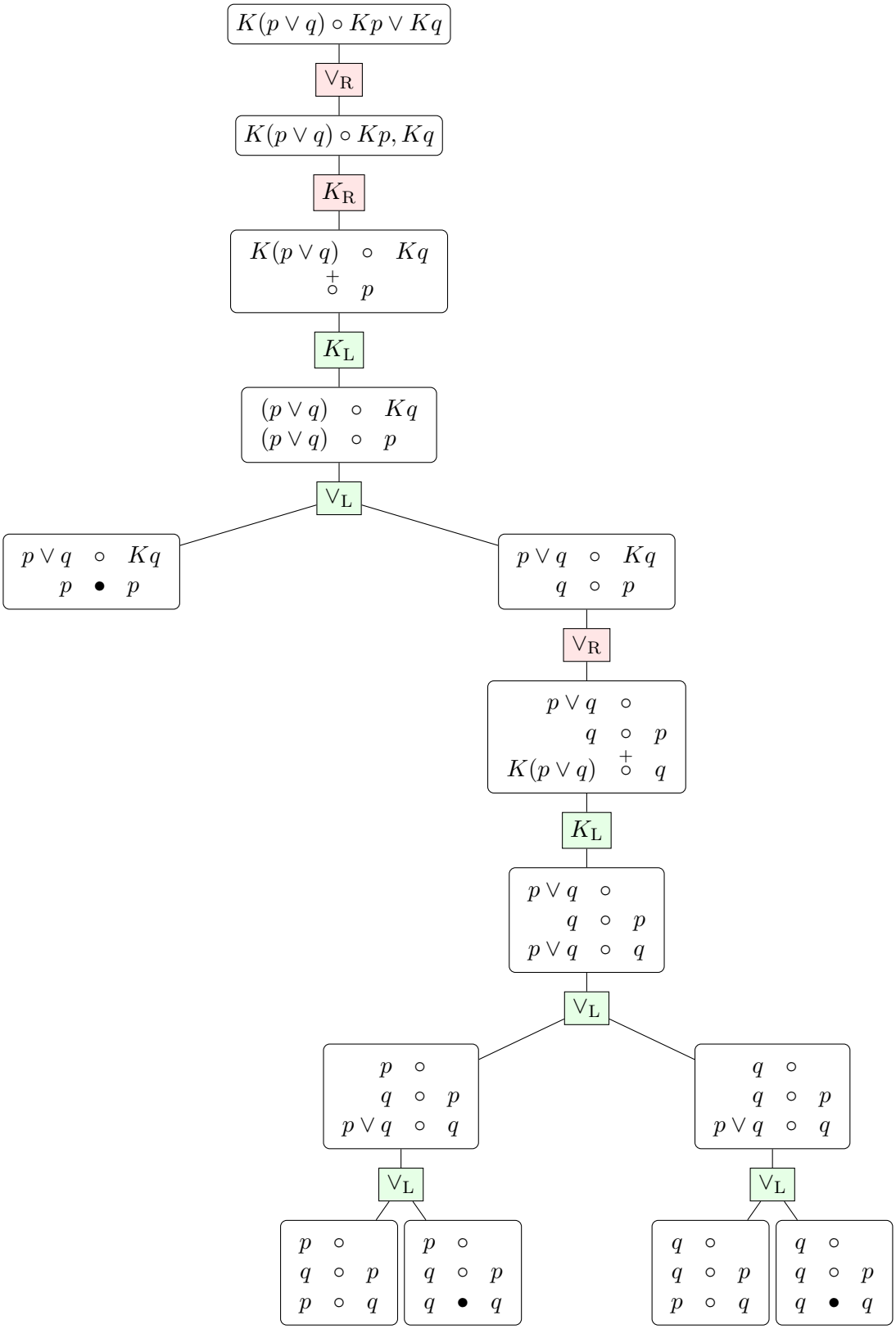


Exercise 8.15 on page 8-27:



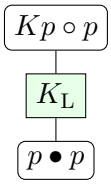
The closed tableau shows that $Kp \vee Kq \models K(p \vee q)$ holds.

Now we show the converse does not hold:

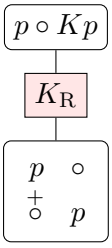

$K(p \vee q) \circ Kp \vee Kq$
$\vee_{\mathrm{R}}$
$K(p \vee q) \circ Kp, Kq$
K_{R}
$K(p \vee q) \circ Kq$
$\overset{+}{\circ}\ p$
K_{L}
$(p \vee q) \circ Kq$
$(p \vee q) \circ p$
$\vee_{\mathrm{L}}$
$p \vee q \circ Kq$
$p \bullet p$
$p \vee q \circ Kq$
$q \circ p$
$\vee_{\mathrm{R}}$
$p \vee q \circ$
$q \circ p$
$K(p \vee q) \overset{+}{\circ} q$
K_{L}
$p \vee q \circ$
$q \circ p$
$p \vee q \circ q$
$\vee_{\mathrm{L}}$
$p \circ$
$q \circ p$
$p \vee q \circ q$
$q \circ$
$q \circ p$
$p \vee q \circ q$
$\vee_{\mathrm{L}}$
$\vee_{\mathrm{L}}$
$p \circ$
$q \circ p$
$p \circ q$
$p \circ$
$q \circ p$
$q \bullet q$
$q \circ$
$q \circ p$
$p \circ q$
$q \circ$
$q \circ p$
$q \bullet q$

We can find two open branches in the right side, showing that $K(p \vee q) \models Kp \vee Kq$ does not hold.

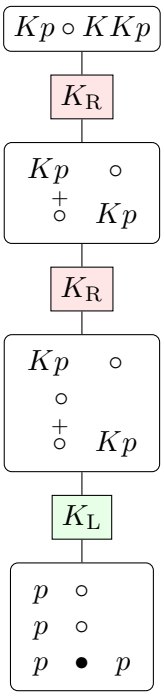
Exercise 8.16 on page 8-28: For $Kp \models p$, we have the following simple table:



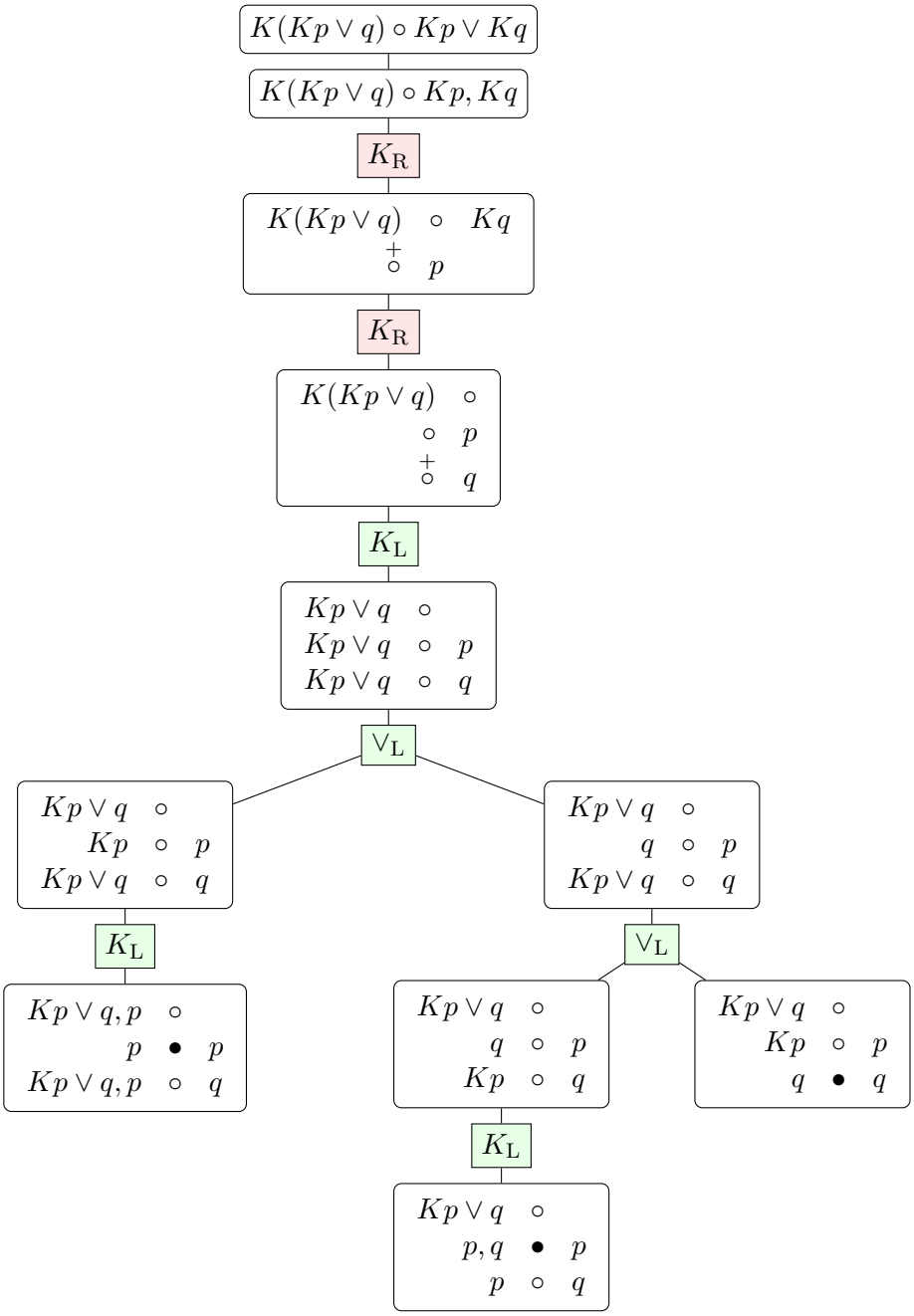
For $p \not\models Kp$, we have a simple open table:



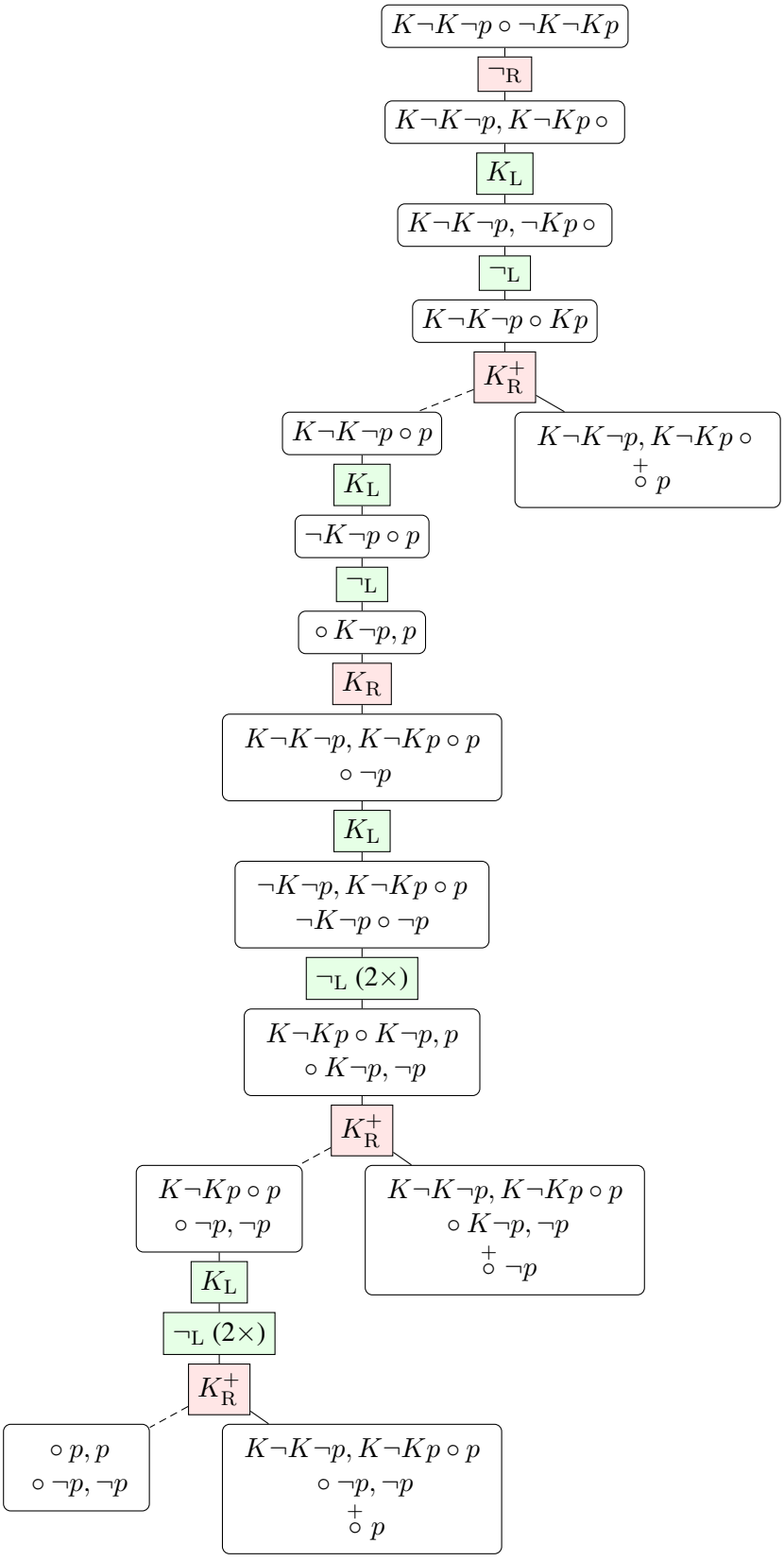
Exercise 8.17 on page 8-28:



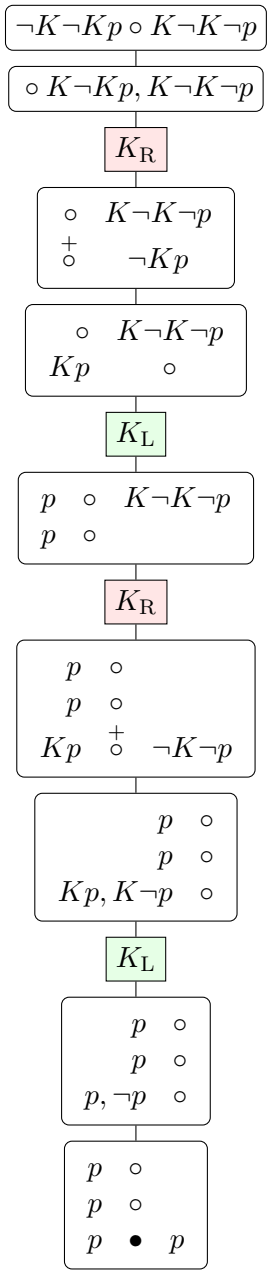
Exercise 8.18 on page 8-28:



Exercise 8.19 on page 8-29:


$K\neg K\neg p \circ \neg K\neg Kp$
$\neg_{\mathrm{R}}$
$K\neg K\neg p, K\neg Kp \circ$
K_{L}
$K\neg K\neg p, \neg Kp \circ$
$\neg_{\mathrm{L}}$
$K\neg K\neg p \circ Kp$
K^{+}_{R}
$K\neg K\neg p \circ p$
$K\neg K\neg p, K\neg Kp \circ \overset{+}{\circ} p$
K_{L}
$\neg K\neg p \circ p$
$\neg_{\mathrm{L}}$
$\circ K\neg p, p$
K_{R}
$K\neg K\neg p, K\neg Kp \circ p \circ \neg p$
K_{L}
$\neg K\neg p, K\neg Kp \circ p \neg K\neg p \circ \neg p$
$\neg_{\mathrm{L}}\ (2\times)$
$K\neg Kp \circ K\neg p, p \circ K\neg p, \neg p$
K^{+}_{R}
$K\neg Kp \circ p \circ \neg p, \neg p$
$K\neg K\neg p, K\neg Kp \circ p \circ K\neg p, \neg p \overset{+}{\circ} \neg p$
K_{L}
$\neg_{\mathrm{L}}\ (2\times)$
K^{+}_{R}
$\circ p, p \circ \neg p, \neg p$
$K\neg K\neg p, K\neg Kp \circ p \circ \neg p, \neg p \overset{+}{\circ} p$

Exercise 8.20 on page 8-30:



Solutions to Exercises from Chapter 9

Exercise 9.1 on page 9-3 First consider the direction from left to right. Suppose we have $\Sigma, \varphi \models \psi$ and $v(\Sigma)$=1. If $v(\varphi)$=1, then it is obvious that $v(\psi)$=1 by the supposition, meaning that $v(\varphi \to \psi)$=1, as required. If $v(\varphi)$=0, then we also have $v(\varphi \to \psi)$=1. Next we prove the direction from

right to left. Suppose that $\Sigma, \varphi \not\models \psi$. Then we have $v(\Sigma)$=1, $v(\varphi)$=1 and $v(\psi)$=0. It follows that $v(\varphi \to \psi)$=0. Hence $\Sigma \not\models \varphi \to \psi$.

Exercise 9.2 on page 9-3 It is not difficult to check that other connectives cannot have both the modus ponens and deduction property. Here we only show $\leftrightarrow$ as an example, other connectives can be done similarly. Now let © be $\leftrightarrow$ and it is possible that φ is false and ψ true in some valuation v. Then we have $\varphi \models \psi$ but $\not\models \varphi \leftrightarrow \psi$, showing that the deduction property is not satisfied.

Exercise 9.3 on page 9-5

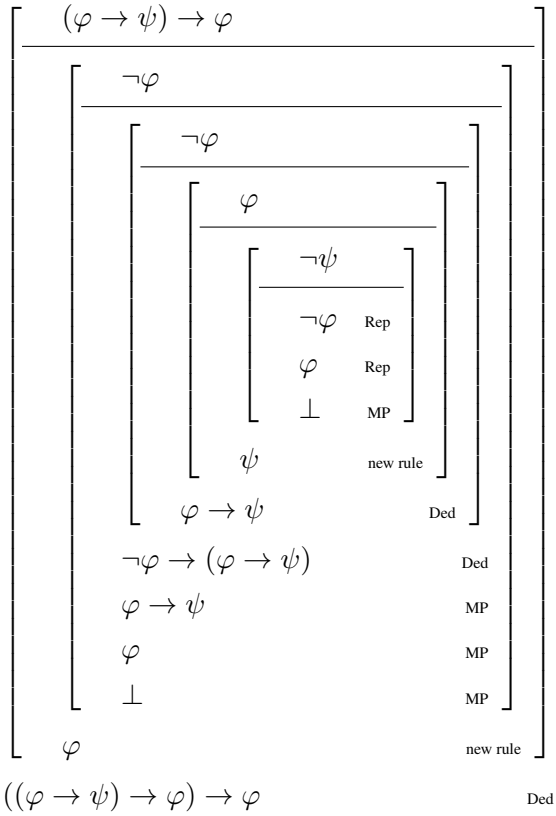
$$\left[\begin{array}{l} \varphi \to (\psi \to \chi) \\ \hline \left[\begin{array}{l} \psi \\ \hline \left[\begin{array}{ll} \varphi & \\ \hline \psi \to \chi & \text{MP} \\ \psi & \text{Rep} \\ \chi & \text{MP} \end{array}\right] \\ \varphi \to \chi \qquad \text{Ded} \end{array}\right] \\ \psi \to (\varphi \to \chi) \qquad \text{Ded} \end{array}\right]$$
$$(\varphi \to (\psi \to \chi)) \to (\psi \to (\varphi \to \chi)) \quad \text{Ded} \tag{B.1}$$

Exercise 9.4 on page 9-6 Please see the following possible proof.

$$\left[\begin{array}{l} \neg\varphi \to \neg\psi \\ \hline \left[\begin{array}{l} \neg\psi \to \bot \\ \hline \left[\begin{array}{ll} \neg\varphi & \\ \hline \neg\psi & \text{MP} \\ \bot & \text{MP} \end{array}\right] \\ \neg\varphi \to \bot \qquad \text{Ded} \end{array}\right] \\ (\neg\psi \to \bot) \to (\neg\varphi \to \bot) \quad \text{Ded} \end{array}\right]$$
$$(\neg\varphi \to \neg\psi) \to ((\neg\psi \to \bot) \to (\neg\varphi \to \bot)) \quad \text{Ded} \tag{B.2}$$

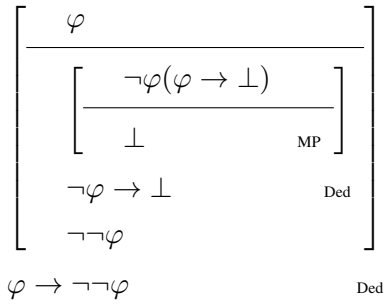
We can get $(\neg\varphi \to \neg\psi) \to (\neg\neg\psi \to \neg\neg\varphi)$ if we abbreviate $\varphi \to \bot$ as $\neg\varphi$. The double negations of $\neg\neg\psi$ and $\neg\neg\varphi$ cannot be eliminated if we only apply modus ponens and deduction rule only.

Exercise 9.5 on page 9-7



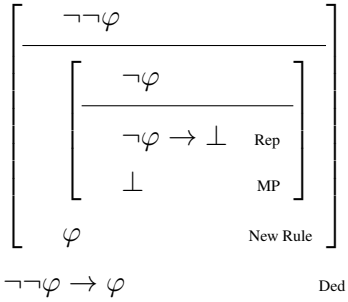
(B.3)

Exercise 9.6 on page 9-7 First we prove $\varphi \to \neg\neg\varphi$.



(B.4)

Next we prove $\neg\neg\varphi \to \varphi$.

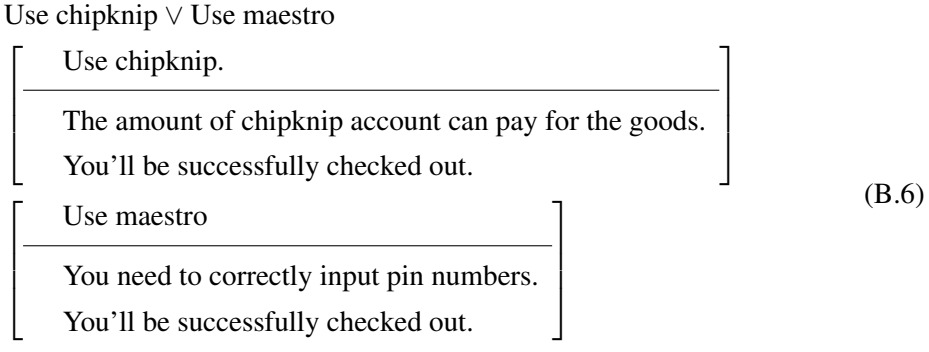


(B.5)

It's clear we have applied the new rule in proving the second.

Exercise 9.7 on page 9-10 First we show that the conclusion holds from left to right. Suppose that $\Sigma, \varphi \vee \psi \models \chi$ and for an arbitrary valuation v which satisfies Σ and φ. It is clear that v also satisfies $\varphi \vee \psi$. Then by supposition, we have v satisfies χ as well. That means $\Sigma, \varphi \models \chi$. Similarly we can get $\Sigma, \psi \models \chi$. Next we show the conclusion holds from right to left. Suppose $\Sigma, \varphi \models \chi$ and $\Sigma, \psi \models \chi$. Consider each valuation v that satisfies Σ and $\varphi \vee \psi$, we need to show that v also satisfies χ. Since $v(\varphi \vee \psi)$=1, we have $v(\varphi)$=1 or $v(\psi)$=1. In either case, we can get $v(\chi)$=1 by supposition. Hence, we conclude that v satisfies χ as well, as required.

Exercise 9.8 on page 9-11 You may consider a scenario of "checkout procedure" in a place where only chipknip and maestro are accepted. Suppose you are buying something in that place and now are to be ready for checkout (the checkout machines in the place are working properly). The goods you selected are probably worth 10 Euros. And you have a Dutch bank debit card (activated and valid) with the amount of checking account more than one thousand Euros, and maestro and chipknip services are provided as well in the bank card, but you are not clear about what is the actual chipknip amount of the bank card. Now you are going to check out first with chipknip. If you have enough amount in the chipknip account for paying the goods, then the machine will show "U HEEFT BETAALD"(You have paid) information meaning that you have successfully checked out. If your chipknip amount is not enough for paying the goods, then you can make checkout via maestro. Commonly you need to input your pin numbers and you know that. It's clearly no problem for you to be successfully check out in that case. Hence, in any case, you will safe for checking out your goods in the place. Here is the deductive representation of the above reasoning:



(B.6)

Exercise 9.9 on page 9-12

$$
\begin{array}{l}
1.\quad \neg\varphi \vee \psi \\
\left[\begin{array}{l} 2.\quad \neg\varphi \\ \hline \left[\begin{array}{l} 3.\quad \varphi \\ \hline 4.\quad \bot \\ 5.\quad \psi \quad \text{E}\bot\ 4 \end{array}\right] \\ 6.\quad \varphi \to \psi \quad \text{Ded} \end{array}\right] \\
\left[\begin{array}{l} 7.\quad \psi \\ \hline 8.\quad \varphi \to \psi \quad \text{I}\to\ 7 \end{array}\right] \\
9.\quad \varphi \to \psi \quad \text{E}\vee\ 1,2\text{-}6,7\text{-}8
\end{array}
\tag{B.7}
$$

Exercise 9.10 on page 9-12

$$
\begin{array}{l}
1.\quad \neg\varphi \vee \neg\psi \\
\left[\begin{array}{l} 2.\quad \neg\varphi \\ \hline \left[\begin{array}{l} 3.\quad \varphi \wedge \psi \\ \hline 4.\quad \varphi \quad \text{E}\wedge \\ 5.\quad \bot \quad \text{E}\neg \end{array}\right] \\ 6.\quad \neg(\varphi \wedge \psi) \quad \text{E}\bot \end{array}\right] \\
\left[\begin{array}{l} 7.\quad \neg\psi \\ \hline \left[\begin{array}{l} 8.\quad \varphi \wedge \psi \\ \hline 9.\quad \psi \quad \text{E}\wedge \\ 10.\quad \bot \quad \text{E}\neg \end{array}\right] \\ 11.\quad \neg(\varphi \wedge \psi) \quad \text{E}\bot \end{array}\right] \\
12.\quad \neg(\varphi \wedge \psi) \quad \text{E}\vee\ 1,2\text{-}6,7\text{-}11
\end{array}
\tag{B.8}
$$

Exercise 9.11 on page 9-12

$$
\begin{array}{ll}
1. & \varphi \vee (\psi \wedge \chi) \\
\left[\begin{array}{lll}
2. & \varphi & \\ \hline
3. & \varphi \vee \psi & \\
4. & \varphi \vee \chi & \\
5. & (\varphi \vee \psi) \wedge (\varphi \vee \chi) & \text{I}\wedge
\end{array}\right] & \\
\left[\begin{array}{lll}
6. & \psi \wedge \chi & \\ \hline
7. & \psi & \\
8. & \varphi \vee \psi & \\
9. & \chi & \\
10. & \varphi \vee \chi & \\
11. & (\varphi \vee \psi) \wedge (\varphi \vee \chi) & \text{I}\wedge
\end{array}\right] & \\
12. & (\varphi \vee \psi) \wedge (\varphi \vee \chi) \quad \text{E}\vee\ 1,2\text{-}5,6\text{-}11
\end{array}
\tag{B.9}
$$

Exercise 9.12 on page 9-12

$$
\begin{array}{ll}
\left[\begin{array}{ll}
\neg((\varphi \to \psi) \vee (\psi \to \varphi)) & \\ \hline
\left[\begin{array}{ll}
\varphi & \\ \hline
\psi \to \varphi & \text{I}\to \\
(\varphi \to \psi) \vee (\psi \to \varphi) & \text{I}\vee \\
\bot &
\end{array}\right] & \\
\left[\begin{array}{ll}
\neg\varphi & \\ \hline
\neg\varphi \vee \psi & \text{I}\vee \\
\varphi \to \psi & \text{Ex.9.9} \\
(\varphi \to \psi) \vee (\psi \to \varphi) & \text{I}\vee \\
\bot &
\end{array}\right] & \\
\bot & \text{E}\vee
\end{array}\right] & \\
(\varphi \to \psi) \vee (\psi \to \varphi) & \text{E}\neg
\end{array}
\tag{B.10}
$$

Exercise 9.13 on page 9-13

$$
\begin{array}{ll}
1. & \neg(\varphi \wedge \psi) \\
\left[\begin{array}{ll}
2. & \neg(\neg\varphi \vee \neg\psi) \\
\hline
\left[\begin{array}{ll} 3. & \neg\varphi \\ \hline 4. & \neg\varphi \vee \neg\psi \quad \text{I}\vee\ 3 \\ 5. & \bot \quad \text{MP } 2,4 \end{array}\right] & \\
6. & \varphi \quad \text{E}\neg\ 3\text{-}5 \\
\left[\begin{array}{ll} 7. & \neg\psi \\ \hline 8. & \neg\varphi \vee \neg\psi \quad \text{I}\vee\ 7 \\ 9. & \bot \quad \text{MP } 2,8 \end{array}\right] & \\
10. & \psi \quad \text{E}\neg\ 7\text{-}9 \\
11. & \varphi \wedge \psi \quad \text{I}\wedge\ 6,10 \\
12. & \bot \quad \text{MP } 1,11
\end{array}\right] & \\
13. & \neg\varphi \vee \neg\psi \quad \text{E}\neg\ 2\text{-}12
\end{array}
\tag{B.11}
$$

Exercise 9.14 on page 9-13

$$
\begin{array}{ll}
1. & \varphi \rightarrow \psi) \\
\left[\begin{array}{ll}
2. & \neg(\neg\varphi \vee \psi) \\
\hline
\left[\begin{array}{ll} 3. & \neg\varphi \\ \hline 4. & \neg\varphi \vee \psi \quad \text{I}\vee\ 3 \\ 5. & \bot \quad \text{MP } 2,4 \end{array}\right] & \\
6. & \varphi \quad \text{E}\neg\ 3\text{-}5 \\
7. & \psi \quad \text{MP } 1,6 \\
8. & \neg\varphi \vee \psi \quad \text{I}\vee\ 7 \\
9. & \bot \quad \text{MP } 2,8
\end{array}\right] & \\
10. & \neg\varphi \vee \psi \quad \text{E}\neg\ 2\text{-}9
\end{array}
\tag{B.12}
$$

Exercise 9.15 on page 9-17

$$
\begin{array}{l}
1.\quad \neg\exists x\, Px \qquad \text{Ass} \\
\left[\begin{array}{l|l}
2. & c \\
\hline
\left[\begin{array}{l|l} 3. & Pc \\ \hline 4. & \exists x\, Px \quad \text{I}\exists\ 3 \\ 5. & \bot \quad \text{MP } 1,4 \end{array}\right] & \\
6.\quad \neg Pc \quad \text{E}\bot\ 3\text{-}5 &
\end{array}\right] \\
7.\quad \forall x\, \neg Px \quad \text{I}\forall\ 2\text{-}6
\end{array}
\tag{B.13}
$$

Exercise 9.16 on page 9-17

$$
\begin{array}{l}
1.\quad \exists x\, (Px \wedge Qx) \qquad \text{Ass} \\
\left[\begin{array}{l|l}
2. & Pc \wedge Qc \qquad c \\
\hline
3. & Pc \\
4. & Qc \\
5. & \exists x\, Px \quad \text{I}\exists\ 3 \\
6. & \exists x\, Qx \quad \text{I}\exists\ 4 \\
7. & \exists x\, Px \wedge \exists x\, Qx
\end{array}\right] \\
8.\quad \exists x\, Px \wedge \exists x\, Qx \quad \text{E}\exists\ 2\text{-}7
\end{array}
\tag{B.14}
$$

Exercise 9.17 on page 9-17

$$
\begin{array}{l}
1.\quad \neg\forall x\, Px \qquad \text{Ass} \\
\left[\begin{array}{l|l}
2. & \neg\exists x\, \neg Px \\
\hline
\left[\begin{array}{l|l} 3. & c \\ \hline \left[\begin{array}{l|l} 4. & \neg Pc \\ \hline 5. & \exists x\, \neg Px \quad \text{I}\exists\ 4 \\ 6. & \bot \quad \text{MP } 2,4 \end{array}\right] & \\ 7. & Pc \quad \text{E}\bot\ 4\text{-}6 \end{array}\right] & \\
8. \quad \forall x\, Px \quad \text{I}\forall\ 3\text{-}7 & 9. \qquad \bot \quad \text{MP } 1,8
\end{array}\right] \\
10.\quad \exists x\, \neg Px \quad \text{E}\bot\ 2\text{-}9
\end{array}
\tag{B.15}
$$

Exercise 9.18 on page 9-17 First we prove that $\exists x\,(Px \vee Qx)$ follows from $\exists x\,Px \vee \exists x\,Qx$.

$$
\begin{array}{ll}
1. & \exists x\,Px \vee \exists x\,Qx \quad \text{Ass} \\
\left[\begin{array}{ll}
2. & \exists x\,Px \\
\hline
\left[\begin{array}{ll}
3. & Pc \qquad\qquad c \\
\hline
4. & Pc \vee Qc \quad \text{I}\vee\ 3 \\
5. & \exists x\,(Px \vee Qx) \quad \text{I}\exists
\end{array}\right] & \\
6. & \exists x\,(Px \vee Qx) \quad \text{E}\exists\ 2\text{-}5
\end{array}\right] & \\
\left[\begin{array}{ll}
7. & \exists x\,Qx \\
\hline
\left[\begin{array}{ll}
8. & Qd \qquad\qquad d \\
\hline
9. & Pd \vee Qd \quad \text{I}\vee\ 8 \\
10. & \exists x\,(Px \vee Qx) \quad \text{I}\exists
\end{array}\right] & \\
11. & \exists x\,(Px \vee Qx) \quad \text{E}\exists\ 7\text{-}10
\end{array}\right] & \\
12. & \exists x\,(Px \vee Qx) \quad \text{E}\vee\ 1\text{-}11
\end{array}
\tag{B.16}
$$

Next we prove the other way around.

$$
\begin{array}{ll}
1. & \exists x\,(Px \vee Qx) \quad \text{Ass} \\
\left[\begin{array}{ll}
2. & Pc \vee Qc \qquad\qquad c \\
\hline
\left[\begin{array}{ll}
3. & Pc \\
\hline
4. & \exists x\,Px \\
5. & \exists x\,Px \vee \exists x\,Qx
\end{array}\right] & \\
\left[\begin{array}{ll}
6. & Qc \\
\hline
7. & \exists x\,Qx \\
8. & \exists x\,Px \vee \exists x\,Qx
\end{array}\right] & \\
9. & \exists x\,Px \vee \exists x\,Qx \quad \text{E}\vee\ 2\text{-}8
\end{array}\right] & \\
10. & \exists x\,Px \vee \exists x\,Qx \quad \text{E}\exists\ 1\text{-}9
\end{array}
\tag{B.17}
$$

Exercise 9.19 on page 9-18

$$
\left[\begin{array}{ll}
1. & \neg\exists x\,(Px \to \forall x\,Px) \\
\hline
2. & \forall x\,\neg(Px \to \forall x\,Px) \quad \text{Ex9.15} \\
3. & \forall x\,(Px \wedge \neg\forall x\,Px) \\
4. & \forall x\,(Px \wedge \exists x\,\neg Px) \quad \text{Ex9.17} \\
\left[\begin{array}{ll} 5. & c \\ \hline 6. & Pc \wedge \exists\neg Px \quad \text{E}\forall \\ 7. & Pc \quad \text{E}\wedge \end{array}\right] & \\
8. & \forall x\,Px \quad \text{I}\forall\ 5\text{-}7 \\
9. & Pc \wedge \exists\neg Px \quad \text{E}\forall \\
10. & \exists\neg Px \quad \text{E}\wedge \\
\left[\begin{array}{lll} 11. & \neg Pd & d \\ \hline 12. & \neg Pd & \end{array}\right] & & \\
13. & \neg Pd \quad \text{E}\exists\ 10\text{-}12 \\
14. & Pd \quad \text{E}\forall\ 8 \\
15. & \bot \quad \text{MP } 13,14
\end{array}\right]
$$

$$
16.\quad \exists x\,(Px \to \forall x\,Px) \quad \text{E}\bot\ 1\text{-}15 \tag{B.18}
$$

Exercise 9.20 on page 9-21

$$
\begin{array}{lll}
1. & \forall x(x \cdot s0 = x \cdot 0 + x) & \text{E}\forall\ \text{P4} \\
2. & \forall x(x \cdot 0) = 0 & \text{P3} \\
3. & \forall x(x \cdot s0 = 0 + x) & \text{E}=\ 1,2 \\
4. & \forall x(0 + x = 0) & \text{Ex9.33} \\
5. & \forall(x \cdot s0 = 0) & \text{E}=\ 3,4
\end{array} \tag{B.19}
$$

Exercise 9.21 on page 9-21

$$
\begin{array}{ll}
1. & \forall x(x \cdot 0) = 0 \quad \text{P3} \\
2. & 0 \cdot 0 = 0 \quad \text{E}\forall\ 1 \\
\left[\begin{array}{ll} 3. & 0 \cdot c = 0c \\ \hline 4. & 0 \cdot sc = 0 \cdot c + 0 \quad \text{E}\forall\ \text{P4} \\ 5. & 0 \cdot sc = 0 + 0 \quad \text{E}=\ 3,4 \\ 6. & 0 + 0 = 0 \quad \text{E}\forall\ \text{P1} \\ 7. & 0 \cdot sc = 0 \quad \text{E}=\ 5,6 \end{array}\right] & \\
8. & \forall x(0 \cdot x = 0) \quad \text{Ind } 2\text{-}7
\end{array} \tag{B.20}
$$

Exercise 9.22 on page 9-22

$$
\left[\begin{array}{lll}
1. & & c \\ \hline
2. & 0+c=c & \text{E}\forall\ 9.33 \\
3. & c+0=c & \text{E}\forall\ \text{P1} \\
4. & c+0=0+c & \text{E}=2,3 \\
 & \left[\begin{array}{lll}
5. & c+d=d+c & d \\ \hline
6. & d+sc=sd+c & \text{E}\forall\ 9.34 \\
7. & c+sd=s(c+d) & \text{E}\forall\ \text{P2} \\
8. & c+sd=s(d+c) & \text{E}=5,7 \\
9. & d+sc=s(d+c) & \text{E}\forall\ \text{P2} \\
10. & c+sd=d+sc & \text{E}=8,9 \\
11. & c+sd=sd+c & \text{E}=6,10
\end{array}\right] & \\
12. & \forall y\,(c+y=y+c) & \text{Ind } 4,5\text{-}11
\end{array}\right]
$$
$$
13.\quad \forall x\forall y\,(x+y=y+x) \qquad \text{I}\forall\ 1\text{-}12 \tag{B.21}
$$

Exercise 9.23 on page 9-22

$$
\begin{array}{lll}
1. & \forall x(x\cdot ss0=x\cdot s0+x) & \text{E}\forall\ \text{P4} \\
2. & \forall x(x\cdot s0=x & \text{Ex9.20} \\
3. & \forall(x\cdot ss0=x+x) & \text{E}=1,2
\end{array} \tag{B.22}
$$

Exercise 9.24 on page 9-22 In order to prove the result $\forall x\forall y(x\cdot y=y\cdot x)$, we first prove two helpful lemmas as (1) $\forall x(s0\cdot x=x)$ and (2) $\forall x\forall y\forall z((x+y)\cdot z=x\cdot z+y\cdot z)$. For (1) we have the following proof:

$$
\begin{array}{lll}
1. & s0\cdot 0=0 & \text{E}\forall\ \text{P3} \\
 & \left[\begin{array}{lll}
2. & s0\cdot c=c & \\ \hline
3. & s0\cdot sc=s0\cdot c+s0 & \text{E}\forall\ \text{P4} \\
4. & s0\cdot sc=c+s0 & \text{E}=2,3 \\
5. & c+s0=sc & \text{E}\forall\ 9.32 \\
6. & s0\cdot sc=sc & \text{E}=4,5
\end{array}\right] & \\
7. & \forall x(s0\cdot x=x) & \text{I}\forall\ 1\text{-}6
\end{array} \tag{B.23}
$$

Next for (2), we may need Ex9.25 (it can be proved independently without Ex9.24) to help proving

this lemma:

$$
\begin{array}{ll}
1. & \forall x \forall y((x+y)\cdot 0 = 0) \quad \text{E}\forall\ \text{P3} \\
2. & \forall x(x \cdot 0 = 0) \quad \text{P3} \\
3. & \forall y(y \cdot 0 = 0) \quad \text{P3} \\
4. & 0 + 0 = 0 \quad \text{E}\forall\ \text{P1} \\
5. & \forall x \forall y((x+y)\cdot 0 = x \cdot 0 + y \cdot 0) \quad \text{E=}\ 1\text{-}4 \\
\left[\begin{array}{ll}
6. & \forall x \forall y((x+y)\cdot c = x \cdot c + y \cdot c) \quad c \\
\hline
7. & \forall x \forall y((x+y)\cdot sc = (x+y)\cdot c + (x+y)) \quad \text{E}\forall\ \text{P4} \\
8. & \forall x \forall y((x+y)\cdot sc = x \cdot c + y \cdot c + (x+y)) \quad \text{E=}\ 6,7 \\
9. & \forall x \forall y((x+y)\cdot sc = (x \cdot c + x) + (y \cdot c + y)) \quad \text{E=}\forall\ \text{Ex9.25 twice} \\
10. & \forall x \forall y((x \cdot c + x) + (y \cdot c + y) = x \cdot sc + y \cdot sc) \quad \text{E}\forall\ \text{P4 twice} \\
11. & \forall x \forall y((x+y)\cdot sc = x \cdot sc + y \cdot sc) \quad \text{E=}\ 9,10
\end{array}\right] \\
12. & \forall x \forall y \forall z((x+y)\cdot z = x \cdot z + y \cdot z) \quad \text{I}\forall\ 5\text{-}11
\end{array}
\tag{B.24}
$$

Now we can prove the final result:

$$
\begin{array}{ll}
\left[\begin{array}{ll}
1. & \quad c \\
\hline
2. & c \cdot 0 = 0 \quad \text{E}\forall\ \text{P3} \\
3. & 0 \cdot c = 0 \quad \text{E}\forall\ \text{Ex9.21} \\
4. & c \cdot 0 = 0 \cdot c \quad \text{E=}\ 2,3 \\
\left[\begin{array}{ll}
5. & c \cdot d = d \cdot c \quad d \\
\hline
6. & c \cdot sd = c \cdot d + c \quad \text{E}\forall\ \text{P4} \\
7. & c \cdot sd = d \cdot c + c \quad \text{E=}\ 5,6 \\
8. & s0 \cdot c = c \quad \text{E}\forall\ \text{Lemma(1)} \\
9. & c \cdot sd = d \cdot c + s0 \cdot c \quad \text{E=}\ 7,8 \\
10. & (d + s0)\cdot c = d \cdot c + s0 \cdot c \quad \text{E}\forall\ \text{Lemma(2)} \\
11. & c \cdot sd = (d + s0)\cdot c \quad \text{E=}\ 9,10 \\
12. & d + s0 = sd \quad \text{E}\forall\ 9.32 \\
13. & c \cdot sd = sd \cdot c \quad \text{E=}\ 11,12
\end{array}\right] \\
14. & \forall y(c \cdot y = y \cdot c) \quad \text{I}\forall\ 4\text{-}13
\end{array}\right] \\
15. & \forall x \forall y(x \cdot y = y \cdot x) \quad \text{I}\forall\ 1\text{-}14
\end{array}
\tag{B.25}
$$

Exercise 9.25 on page 9-22

$$
\begin{array}{lll}
1. & \forall x \forall y((x+y)+0 = x+y) & \text{E}\forall\ \text{P1} \\
2. & \forall y(y+0 = y) & \text{P1} \\
3. & \forall x \forall y(x+y = x+y) & = \\
4. & \forall x \forall y(x+(y+0) = x+y) & \text{E}=\ 2,3 \\
5. & \forall x \forall y(x+(y+0) = (x+y)+0) & \text{E}=\ 1,4 \\
\left[\begin{array}{lll}
6. & \forall x \forall y(x+(y+c) = (x+y)+c) & c \\
\hline
7. & \forall y(y+sc = s(y+c)) & \text{E}\forall\ \text{P2} \\
8. & \forall x \forall y(x+(y+sc) = x+(y+sc)) & = \\
9. & \forall x \forall y(x+(y+sc) = x+s(y+c)) & \text{E}=\ 7,8 \\
10. & \forall x \forall y(x+s(y+c) = s(x+(y+c))) & \text{P2} \\
11. & \forall x \forall y(x+s(y+c) = s((x+y)+c)) & =\ 6,10 \\
12. & \forall x \forall y(s((x+y)+c) = (x+y)+sc) & \text{P2} \\
13. & \forall x \forall y(x+s(y+c) = (x+y)+sc) & =\ 11,12 \\
14. & \forall x \forall y(x+(y+sc) = (x+y)+sc) & =\ 9,13
\end{array}\right] & & \text{(B.26)} \\
15. & \forall x \forall y \forall z(x+(y+z) = (x+y)+z) & \text{I}\forall\ 5\text{-}14
\end{array}
$$

Solutions to Exercises from Chapter 10

Exercise 10.1 on page 10-4:

$$
\begin{array}{lll}
AF & (p \leftrightarrow (q \leftrightarrow r)) = & \\
= & AF((\neg p \vee (q \leftrightarrow r)) \wedge (p \vee \neg(q \leftrightarrow r))) & \\
= & (AF(\neg p \vee (q \leftrightarrow r))) \wedge (AF(p \vee \neg(q \leftrightarrow r))) & \\
= & (AF(\neg p) \vee AF(q \leftrightarrow r)) \wedge (AF(p) \vee AF(\neg(q \leftrightarrow r))) & \\
= & (\neg p \vee AF((\neg q \vee r) \wedge (q \vee \neg r))) \wedge (p \vee \neg AF((\neg q \vee r) \wedge (q \vee \neg r))) & \\
= & (\neg p \vee (AF(\neg q \vee r) \wedge AF(q \vee \neg r))) \wedge (p \vee \neg(AF(\neg q \vee r) \wedge AF(q \vee \neg r))) & \\
= & (\neg p \vee ((AF(\neg q) \vee AF(r)) \wedge (AF(q) \vee AF(\neg r)))) \wedge (p \vee \neg((AF(\neg q) \vee AF(r)) \wedge (AF(q) \vee AF(\neg r)))) & \\
= & (\neg p \vee ((\neg q \vee r) \wedge (q \vee \neg r))) \wedge (p \vee \neg((\neg q \vee r) \wedge (q \vee \neg r))) &
\end{array}
$$

Exercise 10.2 on page 10-5:

$$
\begin{array}{rcl}
NNF(\neg(p \vee \neg(q \wedge r))) & = & NNF(\neg p) \wedge NNF(\neg\neg(q \wedge r)) \\
& = & \neg p \wedge NNF(q \wedge r) \\
& = & \neg p \wedge (NNF(q) \wedge NNF(r)) \\
& = & \neg p \wedge q \wedge r
\end{array}
$$

Exercise 10.3 on page 10-7:

$$\begin{aligned} CNF((p \vee \neg q) \wedge (q \vee r)) &= CNF(p \vee \neg q) \wedge CNF(q \vee r) \\ &= DIST(CNF(p), CNF(\neg q)) \wedge DIST(CNF(q), CNF(r)) \\ &= DIST(p, \neg q) \wedge DIST(q, r) \\ &= (p \vee \neg q) \wedge (q \vee r) \end{aligned}$$

Exercise 10.4 on page 10-7:

$$\begin{aligned} & CNF((p \wedge q) \vee (p \wedge r) \vee (q \wedge r)) = \\ = \; & DIST(CNF(p \wedge q), CNF((p \wedge r) \vee (q \wedge r))) \\ = \; & DIST((CNF(p) \wedge CNF(q)), DIST(CNF(p \wedge r), CNF(q \wedge r))) \\ = \; & DIST((p \wedge q), DIST((CNF(p) \wedge CNF(r)), (CNF(q) \wedge CNF(r)))) \\ = \; & DIST((p \wedge q), DIST((p \wedge r), (q \wedge r))) \\ = \; & DIST((p \wedge q), (DIST(p, (q \wedge r)) \wedge DIST(r, (q \wedge r)))) \\ = \; & DIST((p \wedge q), (DIST(p, q) \wedge DIST(p, r)) \wedge (DIST(r, q) \wedge DIST(r, r))) \\ = \; & DIST((p \wedge q), ((p \vee q) \wedge (p \vee r)) \wedge ((r \vee q) \wedge (r \vee r))) \\ = \; & DIST(p, ((p \vee q) \wedge (p \vee r)) \wedge ((r \vee q) \wedge (r \vee r))) \wedge \\ & \qquad \wedge DIST(q, ((p \vee q) \wedge (p \vee r)) \wedge ((r \vee q) \wedge (r \vee r))) \\ = \; & (DIST(p, ((p \vee q) \wedge (p \vee r))) \wedge DIST(p, ((r \vee q) \wedge (r \vee r)))) \wedge \\ & \qquad \wedge (DIST(q, ((p \vee q) \wedge (p \vee r))) \wedge DIST(q, ((r \vee q) \wedge (r \vee r)))) \\ = \; & (DIST(p, (p \vee q)) \wedge DIST(p, (p \vee r))) \wedge (DIST(p, (r \vee q)) \wedge DIST(p, (r \vee r)))) \wedge \\ & \qquad \wedge (DIST(q, (p \vee q)) \wedge DIST(q, (p \vee r))) \wedge (DIST(q, (r \vee q)) \wedge DIST(q, (r \vee r)))) \\ = \; & (p \vee p \vee q) \wedge (p \vee p \vee r) \wedge (p \vee r \vee q) \wedge (p \vee r \vee r) \wedge (q \vee p \vee q) \wedge (q \vee p \vee r) \wedge (q \vee r \vee q) \wedge (q \vee r \vee r) \end{aligned}$$

Exercise 10.5 on page 10-8:

Assume the premisse is true. Then, because the premise is a clause form $C_1, \ldots, C_i, \ldots, C_n$, every conjunct C_k for $k \in \{1, ..., n\}$ is true. Therefore every C_k for $k \in \{1, \ldots, i-1, i+1, \ldots, n\}$ is true. Hence the conclusion is true and the inference rule is sound.

Exercise 10.6 on page 10-9:

Test the validity of the following inferences using resolution:

(1) $((p \vee q) \wedge \neg q) \rightarrow r, q \leftrightarrow \neg p \models r$

(2) $(p \vee q) \rightarrow r, \neg q, \neg q \leftrightarrow p \models r$

(1) First we translate the inferences in a corresponding clause form as follows:

$$\{\{\neg p, q, r\}, \{\neg p, \neg q\}, \{q, p\}, \{\neg r\}\}$$

next, we apply resolution, to the first and second clauses:

$$\{\{\neg p, r\}, \{q, p\}, \{\neg r\}\}$$

we apply resolution again, to the first and second clauses:

$$\{\{r, q\}, \{\neg r\}\}$$

we apply resolution one more time, to the first and second clauses:

$$\{\{q\}\}$$

The clause form containing the premises and the conclusion negated is satisfiable, therefore the inference is not valid.

(2) First we translate the inferences in a corresponding clause form as follows:

$$\{\{\neg p, r\}, \{\neg q, r\}, \{\neg q\}, \{\neg p, \neg q\}, \{q, p\}, \{\neg r\}\}$$

next, we apply resolution, to the first and fifth clauses:

$$\{\{r, q\}, \{\neg q, r\}, \{\neg q\}, \{\neg p, \neg q\}, \{\neg r\}\}$$

we apply resolution again, to the first and second clauses:

$$\{\{r\}, \{\neg q\}, \{\neg p, \neg q\}, \{\neg r\}\}$$

we apply resolution one more time, to the first and last clauses:

$$\{[], \{\neg q\}, \{\neg p, \neg q\}\}$$

The clause form containing the premises and the conclusion negated is not satisfiable, therefore the inference is valid.

Exercise 10.7 on page 10-9:

Determine which of the following clause forms are satisfiable:

(1) $\{\{\neg p, q\}, \{\neg q\}, \{p, \neg r\}, \{\neg s\}, \{\neg t, s\}, \{t, r\}\}$

(2) $\{\{p, \neg q, r\}, \{q, r\}, \{q\}, \{\neg r, q\}, \{\neg p, r\}\}$

Give a satisfying valuation for the satisfyable case(s).

(1) We start with the clause form:

$$\{\{\neg p, q\}, \{\neg q\}, \{p, \neg r\}, \{\neg s\}, \{\neg t, s\}, \{t, r\}\}.$$

Applying resolution for $\neg q, q$ to the first two clauses gives:

$$\{\{\neg p\}, \{p, \neg r\}, \{\neg s\}, \{\neg t, s\}, \{t, r\}\}.$$

Applying resolution for $\neg p, p$ to the first two clauses gives:

$$\{\{\neg r\}, \{\neg s\}, \{\neg t, s\}, \{t, r\}\}.$$

Applying resolution for $\neg r, r$ to the first and last clauses gives:

$$\{\{\neg s\}, \{\neg t, s\}, \{t\}\}.$$

Applying resolution for $\neg s, s$ to the first two clauses gives:

$$\{\{\neg t\}, \{t\}\}.$$

Applying resolution for $\neg t, t$ to the first two clauses gives:

$$\{[]\}.$$

We have derived a clause form containing the empty clause. We have tried to construct a situation where all clauses are true but this attempt has led us to a contradiction. Hence the clause form is not satisfiable.

(2) We start with the clause form:

$$\{\{p, \neg q, r\}, \{q, r\}, \{q\}, \{\neg r, q\}, \{\neg p, r\}\}$$

Applying resolution for $\neg q, q$ to the first two clauses gives:

$$\{\{p, r\}, \{r\}, \{q\}, \{\neg r, q\}, \{\neg p, r\}\}$$

Applying resolution for $r, \neg r$ to the second and fourth clauses gives:

$$\{\{p, r\}, \{q\}, \{q\}, \{\neg p, r\}\}$$

Applying resolution for $p, \neg p$ to the first and last clauses gives:

$$\{\{r\}, \{q\}, \{q\}, \{r\}\}$$

The clause form is satisfiable, and a valuation that satisfy it is the one in which all propositional atoms p, q and r are true.

Exercise 10.8 on page 10-10:

First we express the constraints as logical formulas, as follows:

- $a \vee b$
- $(a \wedge e \wedge \neg f) \vee (a \wedge \neg e \wedge f) \vee (\neg a \wedge e \wedge f)$
- $b \leftrightarrow c$
- $a \leftrightarrow \neg d$

- $c \leftrightarrow \neg d$
- $\neg d \rightarrow \neg r$

Next we translate each formula in conjunctive normal form, as follows:

- $a \vee b$
- $(a \vee e) \wedge (a \vee f) \wedge (e \vee f) \wedge (\neg a \vee \neg e \vee \neg f)$
- $(\neg b \vee c) \wedge (b \vee \neg c)$
- $(\neg a \vee \neg d) \wedge (a \vee d)$
- $(\neg c \vee \neg d) \wedge (c \vee d)$
- $d \vee \neg e$

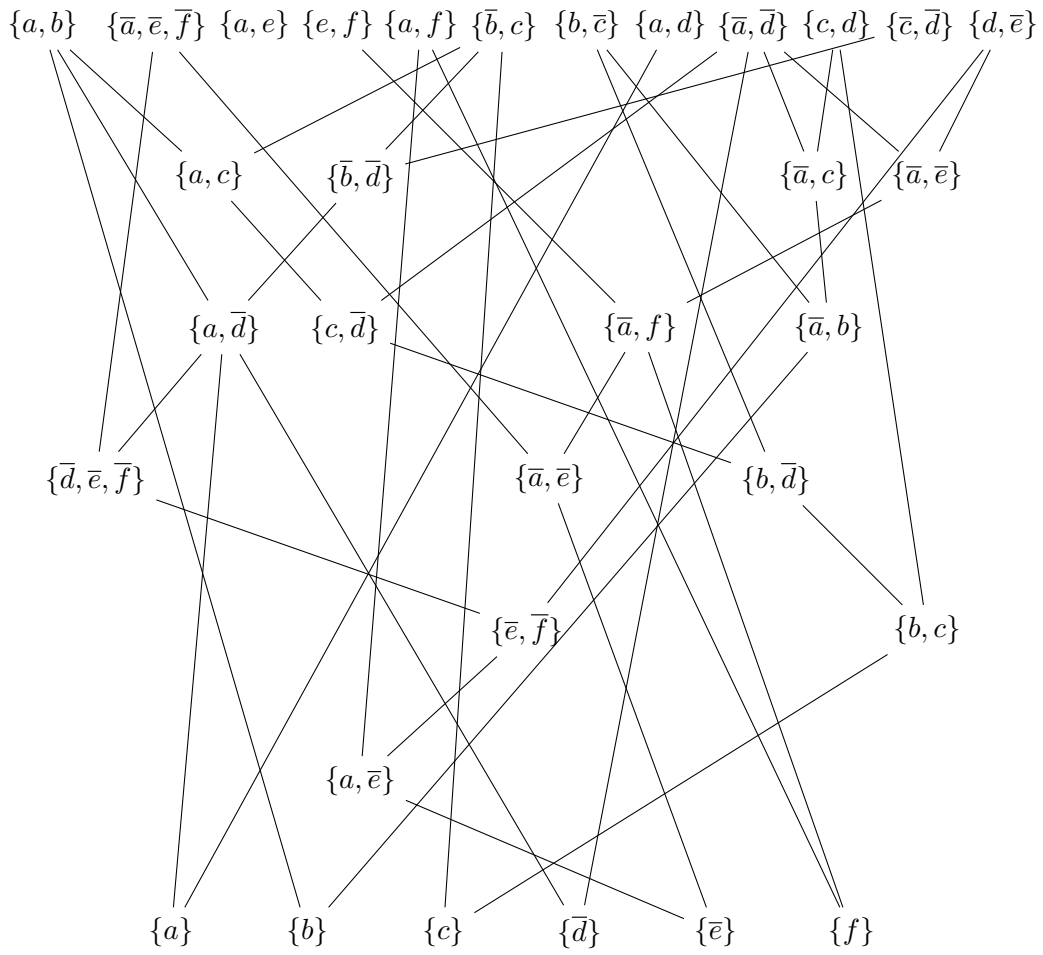
From this we can construct the following clause form:

$$\{\{a, b\}, \{a, e\}, \{a, f\}, \{e, f\}, \{\neg a, \neg e, \neg f\}, \{\neg b, c\}, \{b, \neg c\}, \{\neg a, \neg d\}, \{a, d\}, \{\neg c, \neg d\}, \{c, d\}, \{d, \neg e\}\}$$

Exercise 10.9 on page 10-10:

$$\{\{a, b\}, \{a, e\}, \{a, f\}, \{e, f\}, \{\neg a, \neg e, \neg f\}, \{\neg b, c\}, \{b, \neg c\}, \{\neg a, \neg d\}, \{a, d\}, \{\neg c, \neg d\}, \{c, d\}, \{d, \neg e\}\}$$

We can apply resolution and find out the satisfying valuation in the following way:



Exercise 10.10 on page 10-12:

(1) $\{(1,2),(2,3),(3,4),(1,3),(1,4),(2,4)\}$,

(2) $\{(1,2),(2,3),(3,4),(1,3),(1,4),(2,4)\}$,

(3) $\{(1,2),(2,3),(3,4),(1,3),(1,4),(2,4)\}$,

(4) $\{(1,2),(2,1),(1,1),(2,2)\}$,

(5) $\{(1,1),(2,2)\}$.

Exercise 10.11 on page 10-14: $(\forall x\forall y\forall z((Rxy \wedge Ryz) \rightarrow Rxz) \wedge \forall x\forall y\forall z((Sxy \wedge Syz) \rightarrow Sxz)) \rightarrow (\forall x\forall y\forall z((R \circ Sxy \wedge R \circ Syz) \rightarrow R \circ Sxz))$

Exercise 10.12 on page 10-15:

$$\begin{aligned} R \circ S &= \{(0,2),(2,1),(1,1)\}, \\ (R \circ S)^+ &= \{(0,2),(2,1),(1,1),(0,1)\}. \end{aligned}$$

Solutions to Exercises from Chapter A

Exercise A.1 on page A-2: If every member of a set A is also a member of set B we say that A is a subset of B. But there is no element which is a member of $\emptyset$, the precondition of the definition is false, so the conditional in whole holds vacuously. That is, for every set A, $\emptyset \subseteq A$.

Exercise A.2 on page A-2: Set $\{\emptyset\}$ is a set with a member $\emptyset$, but $\emptyset$ is a set containing no member.

Exercise A.3 on page A-4: It is $\{(n, n+4) \mid n \in \mathbb{N}\}$.

Exercise A.4 on page A-4: We need to show $R \subseteq R^{\smile}$. For every x, y, $(x, y) \in R$, $(y, x) \in R^{\smile}$. Then $(y, x) \in R$ since $R^{\smile} \subseteq R$. It follows that $(x, y) \in R^{\smile}$, as required.

Exercise A.5 on page A-4: (3) and (5).

Exercise A.6 on page A-5: First check the direction from left to right. Suppose R is transitive and for arbitrary x, y, $(x, y) \in R \circ R$. It means that there exists some z such that $(x, z) \in R$ and $(z, y) \in R$. Since R is transitive, we can have that $(x, y) \in R$ as well. Next check the direction from right to left. Suppose $R \circ R \subseteq R$ and for arbitrary x, y, z, $(x, y) \in R$ and $(y, z) \in R$. Then $(x, z) \in R \circ R$. It follows that $(x, z) \in R$ since $R \circ R \subseteq R$, as required.

Exercise A.7 on page A-5: Yes. Please see the relation $<$ on $\mathbb{N}$. It is clear that this relation is transitive but $< \circ <=<$ (over $\mathbb{N}$) does not hold.

Exercise A.8 on page A-5:

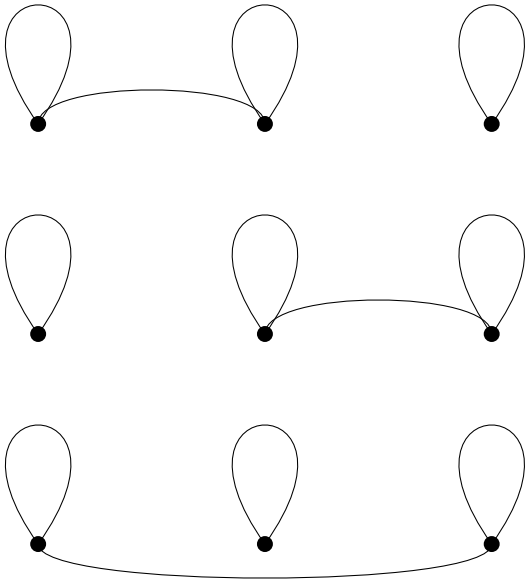
$\{(1,1), (2,2), (3,3), (1,2), (1,3), (2,3)\}$

Exercise A.9 on page A-6:

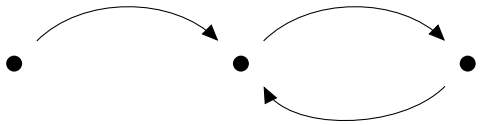
$\{(1,1), (2,2), (3,3), (1,2), (1,3), (2,3), (2,1), (3,1), (3,2)\}$

Exercise A.10 on page A-7: Besides the empty and the universal relations we also have the identity relation:

and also the following, each one with a version in which the isolated point can be not reflexive:



Exercise A.11: Please see the following example.



Exercise A.12: Assume that a relation R is asymmetric, and suppose it contains a loop (x, x). But by the definition of asymmetric, for every x, y, $(x, y) \in R \rightarrow (y, x) \notin R$. It follows $(x, x) \notin R$ since $(x, x) \in R$, a contradiction. So any asymmetric has to be irreflexive.

Exercise A.13 on page A-9: These are the same as the ones given in the solution to exercise A.10, except the empty relation.

Exercise A.14 on page A-9:

R$\not$TS = $\{(2, 1), (1, 2)\}$;

RT$\not$S = $\{(1, 1), (2, 2), (3, 3), (1, 2), (2, 1), (2, 3), (3, 2)\}$;

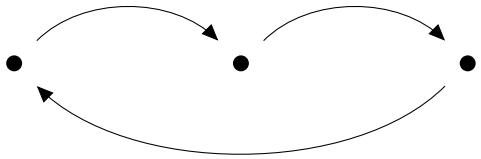
RTS$\not$ = $\{(1, 1), (2, 2), (1, 2)\}$.

What these examples show is that reflexivity does not follow from transitivity and symmetry, that transitivity does not follow from reflexivity and symmetry, and that symmetry does not follow from reflexivity and transitivity.

Exercise A.15 on page A-9: Base case: Proved in the text. Induction Hypothesis: If $|M| < n$ then $\exists x \exists y(Rxy \wedge Ryx)$ or $\exists x \forall y \neg Rxy$ or $\exists x \exists y \exists z(Rxy \wedge Ryx \wedge \neg Rxz)$. Induction step: take

an arbitrary model such that $|M| = n - 1$ and build M' with $dom(M') = dom(M) \cup \{a\}$, such that $R' = R \cup R^a$ where $R^a \subseteq R^A$ with $R^A = \{(x, y) \mid x \in dom(M), y = a\} \cup \{(x, y) \mid y \in dom(M), x = a\} \cup \{(x, y) \mid x = a, y = a\}$. If $\exists x \exists y \exists z(Rxy \wedge Ryx \wedge \neg Rxz)$ then $\exists x \exists y \exists z(R'xy \wedge R'yx \wedge \neg R'xz)$ and we are done. If $\exists x \exists y(Rxy \wedge Ryx)$ then $\exists x \exists y(R'xy \wedge R'yx)$ and we are done. If $\neg\exists x \exists y(Rxy \wedge Ryx)$ and $\neg\exists x \exists y \exists z(Rxy \wedge Ryx \wedge \neg Rxz)$ and $\exists x \forall y \neg Rxy$ then if for some $x \in M$ that has no successor we have $\neg R'xa$ we are done. Suppose that for all $x \in M$ that have no successors we have $R'xa$, then in order for M' to be serial we need to have $R'ay$ for some $y \in M'$. If Raa then asymmetry is violated and we are done. Take an arbitrary $z \in M'$ such that $R'za$ and $z \neq a$ then if we have $R'az$, asymmetry is again violated and we are done. If, for arbitrary $z, y \in M'$ we have $R'za$, $R'ay$ and $y \neq z$ we have to consider two cases. If $\neg Ryz$ then transitivity is violated and we are done. In case Ryz then we have $R'ay$, $R'yz$, and $R'za$. If we satisfy transitivity of R' then we must also have $R'aa$ which in turn violates the asymmetry condition.

Exercise A.16 on page A-9: (1)



(2)

(3)

Exercise A.17 on page A-11: $n \mapsto n + 2$

Exercise A.18 on page A-11: The corresponding characteristic function is $f\colon \mathbb{N}^2 \to \{\mathbf{True}, \mathbf{False}\}$. For every pair $(m, n) \in \mathbb{N}^2$, if $m \leq n$ then $f((m, n)) = \mathbf{True}$, otherwise, $f((m, n)) = \mathbf{false}$.

Exercise A.19 on page A-11: Let $f\colon A \to B$ be a function. Show that the relation $R \subseteq A^2$ given by $(x, y) \in R$ if and only if $f(x) = f(y)$ is an equivalence relation (reflexive, transitive and symmetric) on A. Suppose relation $R \subseteq A^2$ given by $(x, y) \in R$ if and only if $f(x) = f(y)$. We need to show that R is an equivalence relation. First check reflexivity: for every $x \in A$, we have

$f(x) = f(x)$ since f is a function. It follows that $(x, x) \in R$, as required. Next check symmetry: for every pair $(x, y) \in R$, we have $x, y \in A$ and then $f(x) = f(y)$. It's clear $f(y) = f(x)$ meaning that $(y, x) \in R$. Similarly we can check transitivity: for every x, y, zA, $(x, y) \in R$ and $(y, z) \in R$, it follows that $f(x) = f(y)$ and $f(y) = f(z)$ by the premise. Then we have $f(x) = f(z)$. It follows by the same reason that $(x, z) \in R$, as required.

Exercise A.20 on page A-13: We prove that for all natural numbers n:

$$\sum_{k=0}^{n} 2k = n(n+1).$$

Basis. For $n = 0$, we have $\sum_{k=0}^{0} 2k = 0 = 0 \cdot 1$, so for this case the statement holds.

Induction step. We assume the statement holds for some particular natural number n and we show that it also holds for $n + 1$. So assume $\sum_{k=0}^{n} 2k = n(n+1)$. This is the **induction hypothesis**. We have to show: $\sum_{k=0}^{n+1} 2k = (n+1)(n+2)$.

We have:

$$\sum_{k=0}^{n+1} 2k = \sum_{k=0}^{n} 2k + 2(n+1).$$

Now use the induction hypothesis:

$$\sum_{k=0}^{n+1} 2k = \sum_{k=0}^{n} 2k + 2(n+1) \stackrel{ih}{=} n(n+1) + 2(n+1) = n^2 + 3n + 2 = (n+1)(n+2).$$

This settles the induction step. It follows that for all natural numbers n, $\sum_{k=0}^{n} 2k = n(n+1)$.

Bibliography

[Ari89] Aristotle. *Prior Analytics*. Hackett Publishing, Indianapolis IN, 1989. Translated by Robin Smith.

[Axe84] Robert Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.

[BCM+02] F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.

[Bin92] K. Binmore. *Fun and Games*. D.C. Heath, Lexington MA, 1992.

[BN02] F. Baader and W. Nutt. Basic description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logic Handbook*, pages 47–100. Cambridge University Press, 2002.

[Boo54] G. Boole. *An investigation of the laws of thought, on which are founded the mathematical theories of logic and probabilities*. Dover, dover (reprint) edition, 1854.

[Bur98] Stanley N. Burris. *Logic for Mathematics and Computer Science*. Prentice Hall, 1998.

[Cal88] Italo Calvino. *Six Memos for the Next Millennium*. Harvard University Press, 1988.

[Car65] Lewis Carroll. *Alice's Adventures in Wonderland*. MacMillan and Co, 1865.

[CH07] Ian Chiswell and Wilfrid Hodges. *Mathematical Logic*. Oxford Texts in Logic. Oxford University Press, 2007.

[Cla12] Robin Clark. *Meaningful Games: Exploring Language with Game Theory*. MIT Press, Cambridge, Mass and London, England, 2012.

[CSI08] Dan Cryan, Sharron Shatil, and Bill Mayblin (Illustrator). *Logic: A Graphic Guide*. Icon Books., 2008.

[DA28] Hilbert D. and W. Ackermann. *Grundzügen der theoretischen Logik*. Springer, 1928.

[DA50] Hilbert D. and W. Ackermann. *Principles of Mathematical Logic*. American Mathematical Society, translation of the 1928 german edition edition, 1950.

[Dav67] D. Davidson. The logical form of action sentences. In N. Rescher, editor, *The Logic of Decision and Action*, pages 81–95. The University Press, Pittsburgh, 1967.

[DH04] with Yishai Feldman D. Harel. *Algorithmics: The Spirit of Computing*. Pearson Education, third edition edition, 2004.

[DP09] Apostolos Doxiadis and Christos Papadimitriou. *Logicomix: An Epic Search For Truth*. Bloomsbury Publishing, 2009.

[DvdHK06] H.P. van Ditmarsch, W. van der Hoek, and B. Kooi. *Dynamic Epistemic Logic*, volume 337 of *Synthese Library*. Springer, 2006.

[DvE04] K. Doets and J. van Eijck. *The Haskell Road to Logic, Maths and Programming*, volume 4 of *Texts in Computing*. College Publications, London, 2004.

[EV09] J. van Eijck and R. Verbrugge, editors. *Discourses on Social Software*, volume 5 of *Texts in Logic and Games*. Amsterdam University Press, Amsterdam, 2009.

[Fag97] Ronald Fagin. Easier ways to win logical games. In *In Proceedings of the DIMACS Workshop on Finite Models and Descriptive Complexity. American Mathematical Society*, pages 1–32. American Mathematical Society, 1997.

[FHMV95] R. Fagin, J.Y. Halpern, Y. Moses, and M.Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.

[Fre76] Gottlob Frege. *Begriffsschrift, a formula language, modeled upon that of arithmetic, for pure thought*, volume in: From Frege to Gödel (1967), edited by Jean van Heijenoort. Harvard University Press, 1876.

[Fre79] G. Frege. *Begriffsschrift, eine der arithmetischen nachgebildete Formelprache des reinen Denkens*. Verlag Nebert, Halle, 1879.

[Fre67] G. Frege. Begriffsschrift, a formula language, modeled upon that of arithmetic, of pure thought. In J. van Heijenoord, editor, *From Frege to Gödel*, pages 1–82. Harvard University Press, 1967.

[Gam91] L.T.F. Gamut. *Language, Logic and Meaning, Part 1*. Chicago University Press, Chicago, 1991.

[GH09] Steven Givant and Paul Halmos. *Introduction to Boolean Algebras*. Undergraduate Texts in Mathematics. Springer, 2009.

[Har68] Garrett Hardin. The tragedy of the commons. *Science*, 162:1243–48, 1968.

[Hei67] J. van Heijenoort, editor. *From Frege to Gödel: A Source Book in Mathematical Logic, 1879–1931*. Harvard University Press, 19967.

[Hin62] J. Hintikka. *Knowledge and Belief: An Introduction to the Logic of the Two Notions*. Cornell University Press, Ithaca N.Y., 1962.

[HKT00] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic. Foundations of Computing*. MIT Press, Cambridge, Massachusetts, 2000.

[Hoa69] C.A.R. Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):567–580, 583, 1969.

[Hod01] Wilfrid Hodges. *Logic*. Penguin, second edition edition, 2001.

[HR04] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, 2004. Second Edition.

[Jac00] Daniel Jackson. Automating first-order relational logic. *ACM SIGSOFT Software Engineering Notes*, 25(6):130–139, 2000.

[Jac06] Daniel Jackson. *Software Abstractions; Logic, Language and Analysis*. MIT Press, 2006.

[KP81] D. Kozen and R. Parikh. An elementary proof of the completeness of PDL. *Theoretical Computer Science*, 14:113–118, 1981.

[Łuk51] Jan Łukasiewicz. *Aristotle's Syllogistic from the Standpoint of Modern Formal Logic*. Clarendon Press, Oxford, 1951.

[Mat73] Benson Mates. *Stoic Logic*. University of California Press, first edition 1953 edition, 1973.

[Mos08] Lawrence S. Moss. Completeness theorems for syllogistic fragments. In Fritz Hamm and Stephan Kepser, editors, *Logics for Linguistic Structures*, pages 143–174. Mouton de Gruyter, Berlin, New York, 2008.

[MvdH95] J.J.Ch. Meyer and W. van der Hoek. *Epistemic Logic for AI and Computer Science*. Cambridge University Press, 1995.

[NB02] D. Nardi and R. J. Brachman. An introduction to description logics. In F. Baader, D. Calvanese, D.L. McGuinness, D. Nardi, and P.F. Patel-Schneider, editors, *The Description Logic Handbook*, pages 5–44. Cambridge University Press, 2002.

[NM44] John von Neumann and Oscar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[Osb04] Martin J. Osborne. *An Introduction to Game Theory*. Oxford University Press, New York, Oxford, 2004.

[PH91] William Thomas Parry and Edward A. Hacker. *Aristotelean Logic*. State University of New York Press, 1991.

[PH04] I. Pratt-Hartmann. Fragments of language. *Journal of Logic, Language and Information*, 13(2):207–223, 2004.

[Pra78] V. Pratt. A practical decision method for propositional dynamic logic. In *Proceedings 10th Symposium Theory of Computation*, pages 326–337. ACM, 1978.

[Pra80] V. Pratt. Application of modal logic to programming. *Studia Logica*, 39:257–274, 1980.

[Smu09] Raymond M. Smullyan. *The Lady or the Tiger?: and Other Logic Puzzles*. Dover, 2009. First edition: 1982.

[Smu11] Raymond M. Smullyan. *What is the name of this book?* Dover, first edition 1990 edition, 2011.

[Str93] Philip D. Straffin. *Game Theory and Strategy*. The Mathematical Association of America, New Mathematical Library, 1993. Fourth printing: 2002.

[VÏ1] Jouko Väänänen. *Models and Games*. Number 132 in Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2011.

[vB11] J. van Benthem. *Logical Dynamics of Information and Interaction*. Cambridge University Press, 2011.