Chapter 8

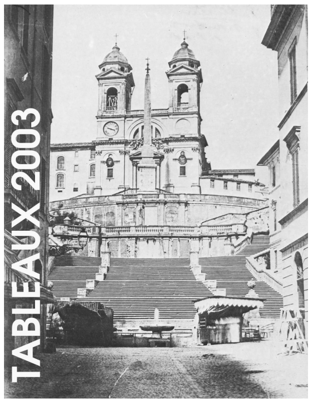
Validity Testing

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.

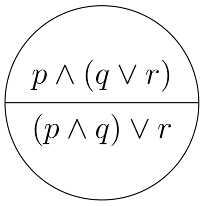
The tables are built up step by step, assigning truth-values to the proposition letters, who represent some atomic bit of propositional information, and then assigning truth-values to all the formulas following the grammatical structures of the formulas. It is therefore called a *bottom up* method.

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:

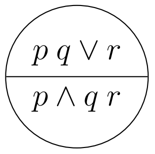
$$p \wedge (q \vee r) \models (p \wedge q) \vee r \tag{8.1}$$

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)

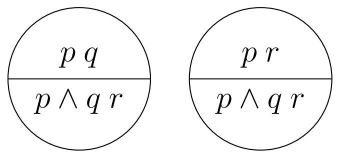
The truth-conditions of the propositions, as defined by the connectives they contain, determine whether this potential counter-example can be realized. As the only true formula is a conjunction, we know that the two conjuncts must be true. The only false proposition is a disjunction, and therefore both these disjuncts must also be false. This leads to the following further specification of our potential counter-example:



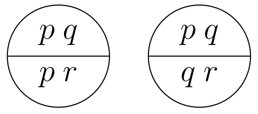
(8.3)

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.



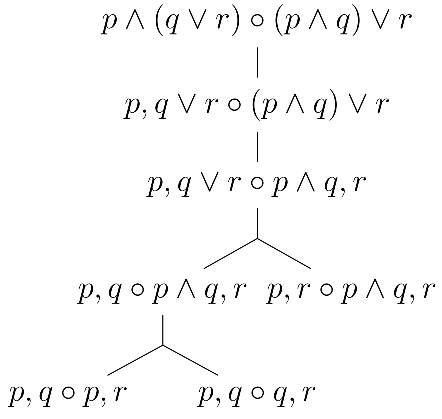
The world on the right hand can not be a counter-example because it requires r to be both true and false. This can never be the case in one single world, and therefore this possibility has to be canceled as a counter-model. The candidate on the left contains a false conjunction, $p \wedge q$. Again, 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.


$p \wedge (q \vee r) \circ (p \wedge q) \vee r$
$p, q \vee r \circ (p \wedge q) \vee r$
$p, q \vee r \circ p \wedge q, r$
$p, q \circ p \wedge q, r$
$p, r \circ p \wedge q, r$
$p, q \circ p, r$
$p, q \circ q, r$

(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.

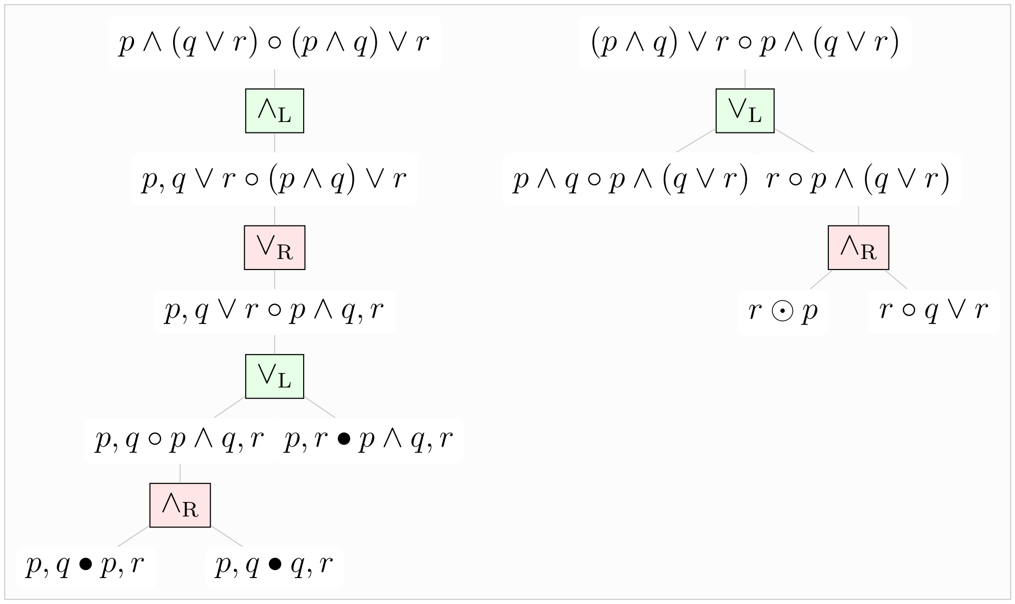
$$\begin{tikzcd}
& (p \wedge q) \vee r \circ p \wedge (q \vee r) \arrow[dl, dash] \arrow[dr, dash] & & \\
p \wedge q \circ p \wedge (q \vee r) & & r \circ p \wedge (q \vee r) \arrow[dl, dash] \arrow[dr, dash] & \\
& r \circ p & & r \circ q \vee r
\end{tikzcd} \tag{8.6}$$

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

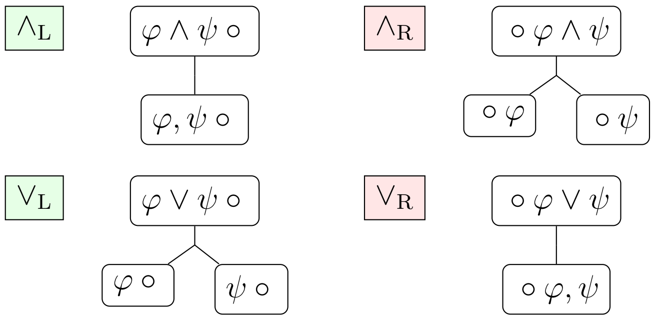
A proper tableau needs to be set up according precise *reduction rules*. A reduction rule is specified by the logical symbol that is to be removed, and the truth-value of the formula as indicated by the sequent (left or right of the truth-falsity separation symbol $\circ$). Such a rule is defined by 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



$p \wedge (q \vee r) \circ (p \wedge q) \vee r$
$\wedge_{\mathrm{L}}$
$p, q \vee r \circ (p \wedge q) \vee r$
$\vee_{\mathrm{R}}$
$p, q \vee r \circ p \wedge q, r$
$\vee_{\mathrm{L}}$
$p, q \circ p \wedge q, r$
$p, r \bullet p \wedge q, r$
$\wedge_{\mathrm{R}}$
$p, q \bullet p, r$
$p, q \bullet q, r$
$(p \wedge q) \vee r \circ p \wedge (q \vee r)$
$\vee_{\mathrm{L}}$
$p \wedge q \circ p \wedge (q \vee r)$
$r \circ p \wedge (q \vee r)$
$\wedge_{\mathrm{R}}$
$r \odot p$
$r \circ q \vee r$

Figure 8.1: Complete tableaus for the earlier examples.

examples.

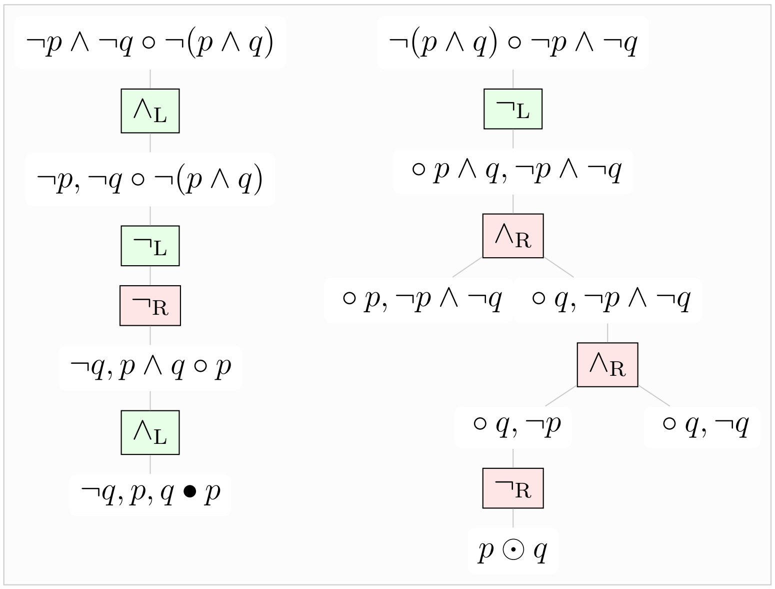


(8.7)

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.

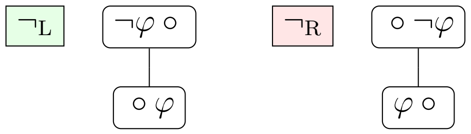
For the other connectives rules can be given quite straightforwardly by using the truth-conditions which have been defined in the introductory chapter on propositional logic.



$\neg p \wedge \neg q \circ \neg(p \wedge q)$
$\wedge_L$
$\neg p, \neg q \circ \neg(p \wedge q)$
$\neg_L$
$\neg_R$
$\neg q, p \wedge q \circ p$
$\wedge_L$
$\neg q, p, q \bullet p$
$\neg(p \wedge q) \circ \neg p \wedge \neg q$
$\neg_L$
$\circ\, p \wedge q, \neg p \wedge \neg q$
$\wedge_R$
$\circ\, p, \neg p \wedge \neg q$
$\circ\, q, \neg p \wedge \neg q$
$\wedge_R$
$\circ\, q, \neg p$
$\circ\, q, \neg q$
$\neg_R$
$p \odot q$

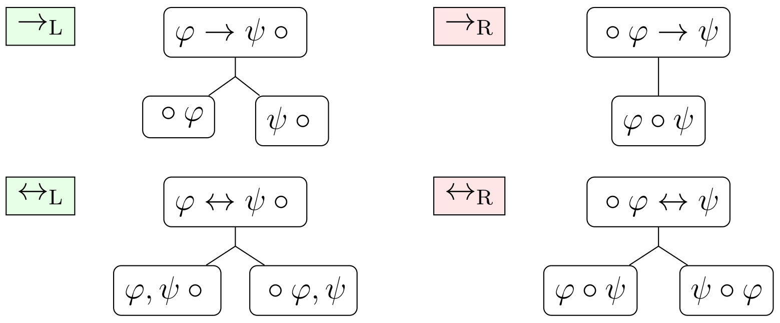
Figure 8.2: Two tableaus with negations. The left tableau shows that $\neg p \wedge \neg q \models \neg(p \wedge q)$. The right tableau shows that the converse of this inference is not valid $\neg(p \wedge q) \not\models \neg p \wedge \neg q$. The counter-model which has been found in the open branch 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)

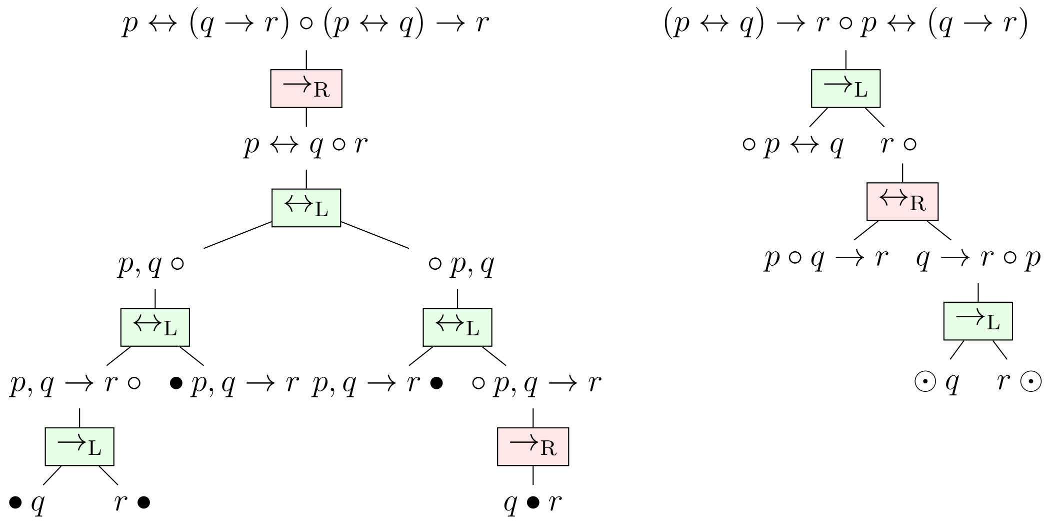
The rules for equivalence are quite easy to understand. An equivalence is true if the truth-values of the two arguments are the same. In terms of reductions this means that if the equivalence appear on the left hand side of the sequent the two arguments remain on the left hand side (both true) or they 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.



$p \leftrightarrow (q \to r) \circ (p \leftrightarrow q) \to r$
$\to_{\mathrm{R}}$
$p \leftrightarrow q \circ r$
$\leftrightarrow_{\mathrm{L}}$
$p, q \circ$
$\circ\, p, q$
$\leftrightarrow_{\mathrm{L}}$
$\leftrightarrow_{\mathrm{L}}$
$p, q \to r \circ$
$\bullet\, p, q \to r$
$p, q \to r \bullet$
$\circ\, p, q \to r$
$\to_{\mathrm{L}}$
$\to_{\mathrm{R}}$
$\bullet\, q$
$r \bullet$
$q \bullet r$
$(p \leftrightarrow q) \to r \circ p \leftrightarrow (q \to r)$
$\to_{\mathrm{L}}$
$\circ\, p \leftrightarrow q$
$r \circ$
$\leftrightarrow_{\mathrm{R}}$
$p \circ q \to r$
$q \to r \circ p$
$\to_{\mathrm{L}}$
$\odot\, q$
$r \odot$

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

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) $\{p \leftrightarrow (q \vee r), \neg q \rightarrow \neg r, \neg(q \wedge p), \neg p\}$

(2) $\{p \vee q, \neg(p \rightarrow q), (p \wedge q) \leftrightarrow p\}$

Exercise 8.5 Check, using the tableau method, whether the following formulas are tautologies or not.

(1) $(p \rightarrow q) \vee (q \rightarrow p)$

(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:

$$\begin{tikzcd}[row sep=small] \forall x\,(Px \vee Qx) \circ \forall x\,Px \vee \forall x\,Qx \arrow[d, dash] \\ \boxed{\vee_{\mathrm{R}}} \arrow[d, dash] \\ \forall x\,(Px \vee Qx) \circ \forall x\,Px, \forall x\,Qx \end{tikzcd} \tag{8.12}$$

[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 $P \lor Q$-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:

$$\begin{tikzcd}[row sep=small] \forall x\,(Px \lor Qx) \circ \forall x\,Px, \forall x\,Qx \arrow[d, dash] \\ \boxed{\forall_{\mathrm{R}}} \arrow[d, dash] \\ \forall x\,(Px \lor Qx) \circ Pd_1, \forall x\,Qx \end{tikzcd} \tag{8.13}$$

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]

$$\begin{tikzcd}[row sep=small] \forall x\,(Px \lor Qx) \circ Pd_1, \forall x\,Qx \arrow[d, dash] \\ \boxed{\forall_{\mathrm{R}}} \arrow[d, dash] \\ \forall x\,(Px \lor Qx) \circ Pd_1, Qd_2 \end{tikzcd} \tag{8.14}$$

At this stage we have to eliminate the universal quantifier on the left hand side of the sequent. We need to apply the property $Px \lor Qx$ 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 $Pd_1 \lor Qd_1$ and $Pd_2 \lor Qd_2$. 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.

$$\begin{tikzcd}[row sep=small] \forall x\,(Px \lor Qx) \circ Pd_1, Qd_2 \arrow[d, dash] \\ \boxed{\forall_{\mathrm{L}}} \arrow[d, dash] \\ Pd_1 \lor Qd_1, Pd_2 \lor Qd_2 \circ Pd_1, Qd_2 \end{tikzcd} \tag{8.15}$$

[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.

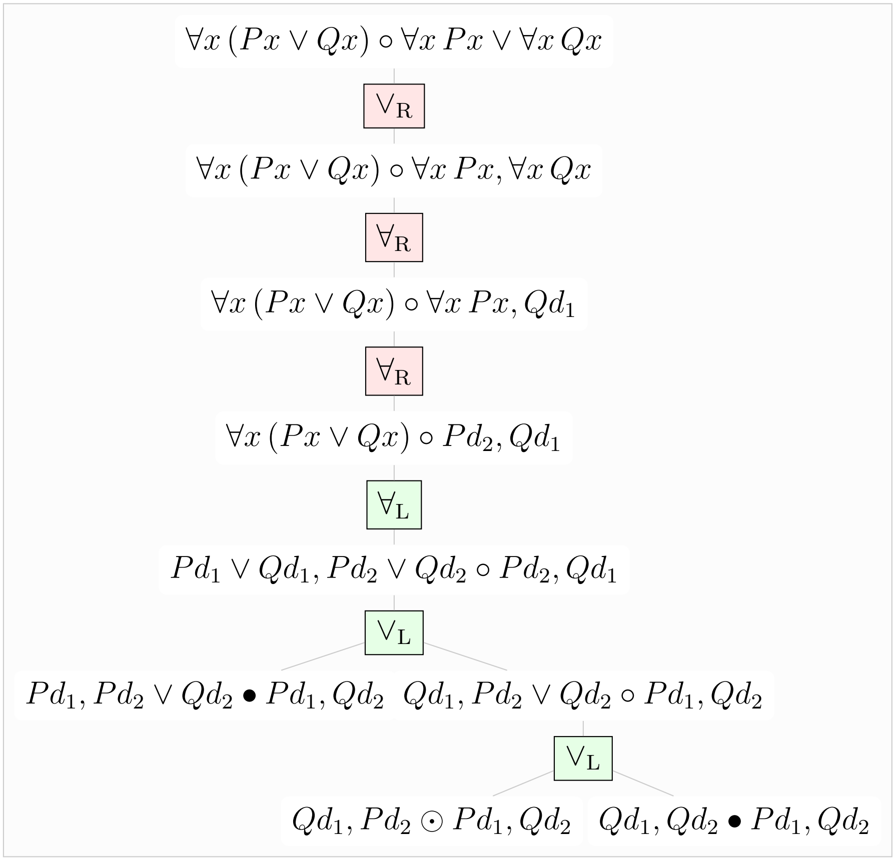

$\forall x\,(Px \vee Qx) \circ \forall x\, Px \vee \forall x\, Qx$
$\vee_{\rm R}$
$\forall x\,(Px \vee Qx) \circ \forall x\, Px, \forall x\, Qx$
$\forall_{\rm R}$
$\forall x\,(Px \vee Qx) \circ \forall x\, Px, Qd_1$
$\forall_{\rm R}$
$\forall x\,(Px \vee Qx) \circ Pd_2, Qd_1$
$\forall_{\rm L}$
$Pd_1 \vee Qd_1, Pd_2 \vee Qd_2 \circ Pd_2, Qd_1$
$\vee_{\rm L}$
$Pd_1, Pd_2 \vee Qd_2 \bullet Pd_1, Qd_2$
$Qd_1, Pd_2 \vee Qd_2 \circ Pd_1, Qd_2$
$\vee_{\rm L}$
$Qd_1, Pd_2 \odot Pd_1, Qd_2$
$Qd_1, Qd_2 \bullet Pd_1, Qd_2$

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.

$$\begin{tikzcd}
 & Pd_1 \vee Qd_1, Pd_2 \vee Qd_2 \circ Pd_1, Qd_2 \arrow[d, -] & & \\
 & \boxed{\vee_{\rm L}} \arrow[dl, -] \arrow[dr, -] & & \\
Pd_1, Pd_2 \vee Qd_2 \bullet Pd_1, Qd_2 & & Qd_1, Pd_2 \vee Qd_2 \circ Pd_1, Qd_2 \arrow[d, -] & \\
 & & \boxed{\vee_{\rm L}} \arrow[dl, -] \arrow[dr, -] & \\
 & Qd_1, Pd_2 \odot Pd_1, Qd_2 & & Qd_1, Qd_2 \bullet Pd_1, Qd_2
\end{tikzcd} \tag{8.16}$$

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.

$$\begin{tikzcd}
\circ\; \exists x\,(Px \to \forall y\, Py) \arrow[d, dash] \\
\boxed{\exists_{\mathrm{R}}} \arrow[d, dash] \\
\circ\; Pd_1 \to \forall y\, Py \arrow[d, dash] \\
\boxed{\to_{\mathrm{R}}} \arrow[d, dash] \\
Pd_1 \circ \forall y\, Py \arrow[d, dash] \\
\boxed{\forall_{\mathrm{R}}} \arrow[d, dash] \\
Pd_1 \circ Pd_2
\end{tikzcd} \tag{8.18}$$

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:

$$\begin{array}{c} Pd_1 \circ \forall y Py \\ | \\ \boxed{\forall_{\mathrm{R}}} \\ | \\ Pd_1 \circ Pd_2, \exists x\,(Px \to \forall y\, Py) \end{array} \tag{8.19}$$

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.

$$\begin{array}{c} Pd_1 \circ Pd_2, \exists x\,(Px \to \forall y\, Py) \\ | \\ \boxed{\exists_{\mathrm{R}}} \\ | \\ Pd_1 \circ Pd_2, Pd_2 \to \forall y\, Py \end{array} \tag{8.20}$$

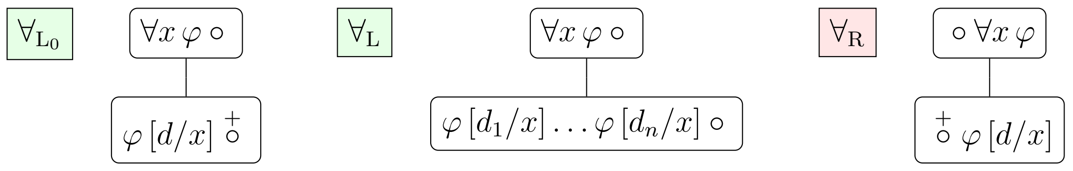
One may think, at first sight, that this leads to an infinite procedure. In this case, things work out nicely, since the tableau closes in the next step. The implication will be removed, and then we run into a conflict: Pd_2 must be true and false at the same time.

$$\begin{array}{c} Pd_1 \circ Pd_2, Pd_2 \to \forall y\, Py \\ | \\ \boxed{\to_{\mathrm{R}}} \\ | \\ Pd_1, Pd_2 \bullet Pd_2, \forall y\, Py \end{array} \tag{8.21}$$

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.


$\forall_{L_0}$
$\forall x\,\varphi \circ$
$\varphi\,[d/x] \overset{+}{\circ}$
$\forall_L$
$\forall x\,\varphi \circ$
$\varphi\,[d_1/x] \ldots \varphi\,[d_n/x] \circ$
$\forall_R$
$\circ\, \forall x\,\varphi$
$\overset{+}{\circ}\, \varphi\,[d/x]$

(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.

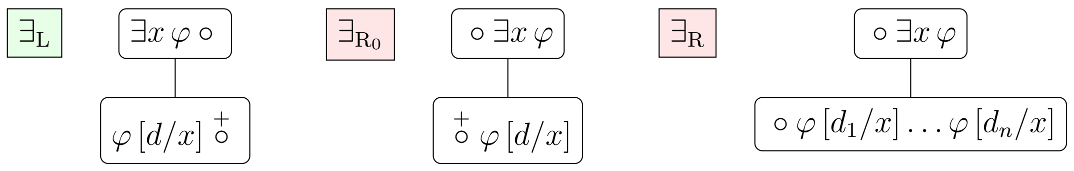
$\forall_L$: If names are present in the input sequent then $\forall x\,\varphi$ can be removed from the left part of the sequent by applying φ to the names $d_1, ..., d_n$, all occurring in the sequent and which φ has *not* been applied to *yet*.

$\forall_R$: 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:


$\exists_L$
$\exists x\,\varphi \circ$
$\varphi\,[d/x] \overset{+}{\circ}$
$\exists_{R_0}$
$\circ\, \exists x\,\varphi$
$\overset{+}{\circ}\, \varphi\,[d/x]$
$\exists_R$
$\circ\, \exists x\,\varphi$
$\circ\, \varphi\,[d_1/x] \ldots \varphi\,[d_n/x]$

(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.

$$\begin{tikzcd}[row sep=small]
\exists y \forall x\, Rxy \circ \forall x\, \exists y\, Rxy \arrow[d, dash] \\
\boxed{\exists_{\mathrm{L}}} \arrow[d, dash] \\
\forall x\, Rxd_1 \overset{+}{\circ} \forall x\, \exists y\, Rxy \arrow[d, dash] \\
\boxed{\forall_{\mathrm{R}}} \arrow[d, dash] \\
\forall x\, Rxd_1 \overset{+}{\circ} \exists y\, Rd_2y \arrow[d, dash] \\
\boxed{\forall_{\mathrm{L}}} \arrow[d, dash] \\
Rd_1d_1, Rd_2d_1 \circ \exists y\, Rd_2y \arrow[d, dash] \\
\boxed{\exists_{\mathrm{R}}} \arrow[d, dash] \\
Rd_1d_1, Rd_2d_1 \bullet Rd_2d_1, Rd_2d_2
\end{tikzcd} \tag{8.24}$$

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.

$$\begin{tikzcd}[row sep=small]
\forall x\, \exists y\, Rxy \circ \exists y \forall x\, Rxy \arrow[d, dash] \\
\boxed{\forall_{\mathrm{L}_0}} \arrow[d, dash] \\
\exists y\, Rd_1y \overset{+}{\circ} \exists y \forall x\, Rxy \arrow[d, dash] \\
\boxed{\exists_{\mathrm{L}}} \arrow[d, dash] \\
\forall x \exists y\, Rxy, Rd_1d_2 \overset{+}{\circ} \exists y \forall x\, Rxy
\end{tikzcd} \tag{8.25}$$

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:

$$\begin{array}{c}
\forall x\, \exists y\, Rxy, Rd_1 d_2 \circ \exists y \forall x\, Rxy \\
| \\
\boxed{\exists_{\mathrm{R}}} \\
| \\
\forall x\, \exists y\, Rxy, Rd_1 d_2 \circ \forall x\, Rxd_1, \forall x\, Rxd_2 \\
| \\
\boxed{\forall_{\mathrm{R}}} \\
| \\
\forall x\, \exists y\, Rxy, Rd_1 d_2 \overset{+}{\circ} Rd_3 d_1, \forall x\, Rxd_1, \exists y \forall x\, Rxy
\end{array} \tag{8.26}$$

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) $\neg\exists x\,(Ax \wedge Bx), \forall x\,(Bx \rightarrow Cx) / \neg\exists x\,(Cx \wedge Ax)$

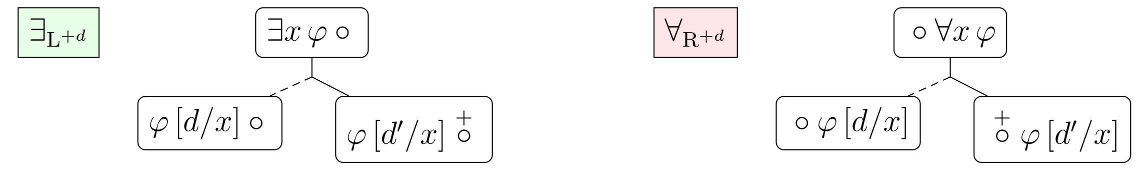
Exercise 8.10 Prove the validity of the following inference with tableaus:

(1) $\forall x\,(Ax \rightarrow Bx) \vee \forall y\,(By \rightarrow Ay) \models \forall x \forall y\,((Ax \wedge By) \rightarrow (Bx \vee Ay))$

(2) $\forall x \forall y\,((Ax \wedge By) \rightarrow (Bx \vee Ay)) \models \forall x\,(Ax \rightarrow Bx) \vee \forall y\,(By \rightarrow Ay)$

8.2.2 Alternative rules for finding finite counter-models

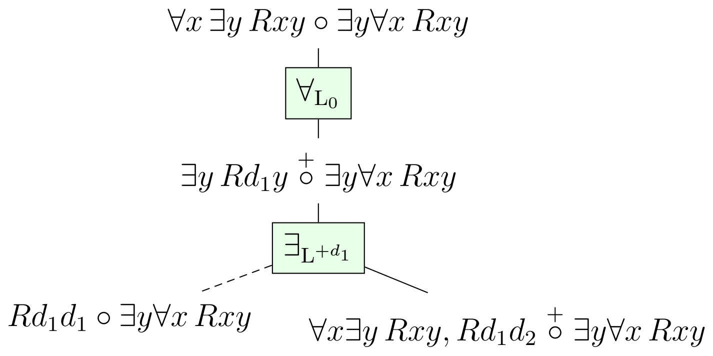
In (8.25) and (8.26) we have seen an example of an invalid inference with quite simple finite counter-models which can not be found by means of the rules for the quantifiers. In order to find such finite counter-models with a tableau system we need to extend the rules for the quantifiers a bit. The problem 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.


$\exists_{\mathrm{L}+d}$
$\exists x\,\varphi \circ$
$\varphi\,[d/x] \circ$
$\varphi\,[d'/x] \overset{+}{\circ}$
$\forall_{\mathrm{R}+d}$
$\circ \forall x\,\varphi$
$\circ\, \varphi\,[d/x]$
$\overset{+}{\circ}\, \varphi\,[d'/x]$

(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.


$\forall x\,\exists y\,Rxy \circ \exists y \forall x\,Rxy$
$\forall_{\mathrm{L}_0}$
$\exists y\,Rd_1y \overset{+}{\circ} \exists y \forall x\,Rxy$
$\exists_{\mathrm{L}+d_1}$
$Rd_1d_1 \circ \exists y \forall x\,Rxy$
$\forall x \exists y\,Rxy, Rd_1d_2 \overset{+}{\circ} \exists y \forall x\,Rxy$

(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_1y 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.

$$\begin{tikzcd}
& Rd_1d_1 \circ \exists y \forall x\, Rxy \arrow[d, dash] & \\
& \boxed{\exists_{\mathrm{R}}} \arrow[d, dash] & \\
& Rd_1d_1 \circ \forall x\, Rxd_1 \arrow[d, dash] & \\
& \boxed{\forall_{\mathrm{R}}} \arrow[d, dash] & \\
& \forall x \exists y\, Rxy, Rd_1d_1 \overset{+}{\circ} Rd_2d_1, \exists y \forall x\, Rxy \arrow[d, dash] & \\
& \boxed{\forall_{\mathrm{L}}} \arrow[d, dash] & \\
& \exists y\, Rd_2y, Rd_1d_1 \circ Rd_2Rd_1, \exists y \forall x\, Rxy \arrow[d, dash] & \\
& \boxed{\exists_{\mathrm{L}^{+d_2}}} \arrow[dl, dash, dashed] \arrow[dr, dash] & \\
Rd_2d_2, Rd_1d_1 \circ Rd_2d_1, \exists y \forall x\, Rxy & & \forall x \exists y\, Rxy, Rd_2d_3, Rd_1d_1 \overset{+}{\circ} Rd_2d_1, \exists y \forall x\, Rxy
\end{tikzcd} \tag{8.29}$$

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.

$$\begin{tikzcd}
& Rd_2d_2, Rd_1d_1 \circ Rd_2d_1, \exists y \forall x\, Rxy \arrow[d, dash] & \\
& \boxed{\exists_{\mathrm{R}}} \arrow[d, dash] & \\
& Rd_2d_2, Rd_1d_1 \circ Rd_2d_1, \forall x\, Rxd_2 \arrow[d, dash] & \\
& \boxed{\forall_{\mathrm{R}^{+d_1}}} \arrow[dl, dash, dashed] \arrow[dr, dash] & \\
Rd_2d_2, Rd_1d_1 \odot Rd_2d_1, Rd_1d_2 & & \forall x\, \exists y\, Rxy \circ Rd_3d_2, \exists y \forall x\, Rxy
\end{tikzcd} \tag{8.30}$$

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. (8.32)

There is at least a pair of persons who love each other.

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:

$$\exists x \forall y\,(Rxy \leftrightarrow \neg\exists z\,(Ryz \wedge Rzy)) \tag{8.33}$$

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 $\exists x \forall y\,(Ryx \leftrightarrow \neg Ryy)$ 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 $\neg\exists z\,(Ryz \wedge Rzy)$ 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.

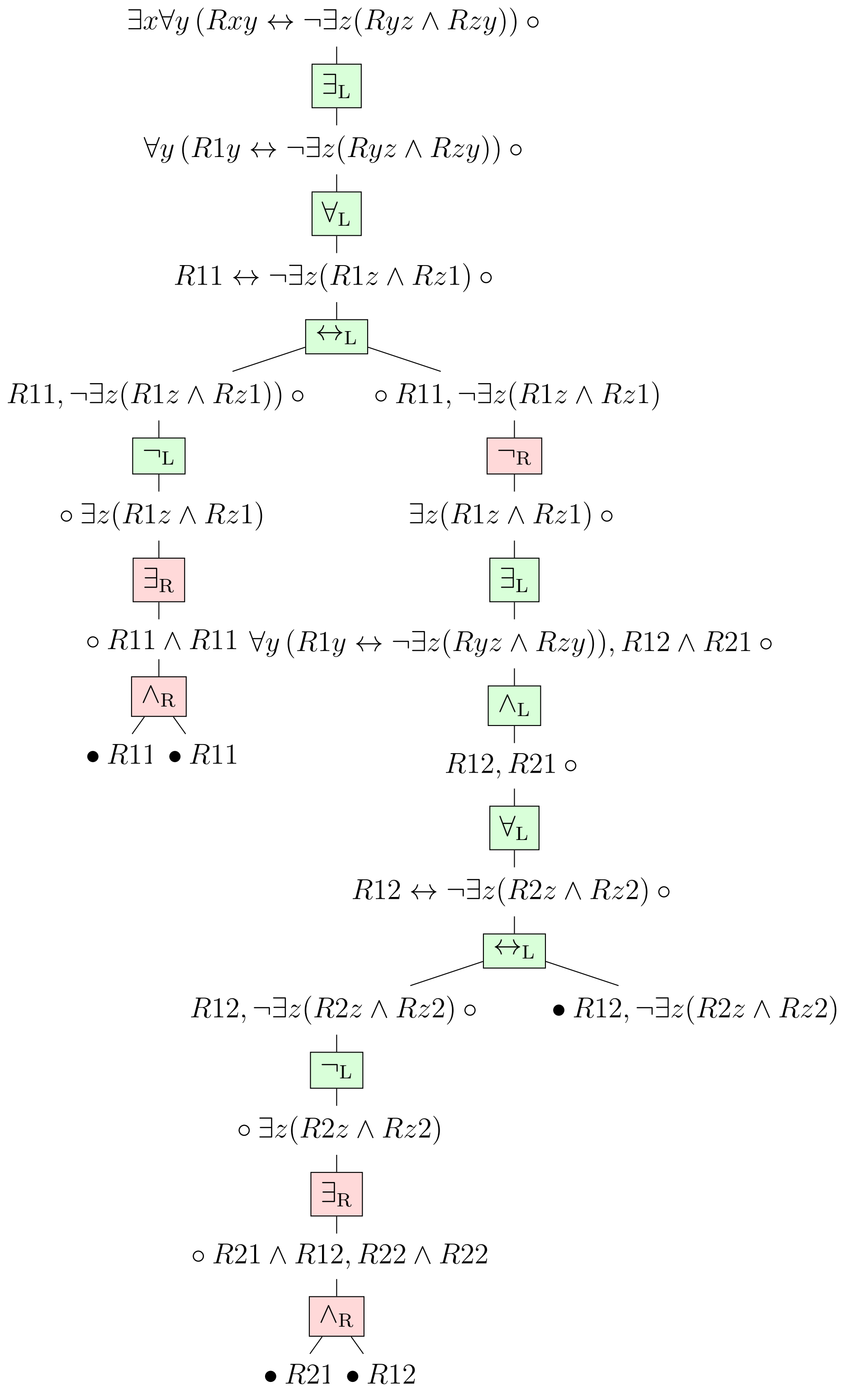

$\exists x \forall y\,(Rxy \leftrightarrow \neg\exists z(Ryz \wedge Rzy))\ \circ$
$\exists_L$
$\forall y\,(R1y \leftrightarrow \neg\exists z(Ryz \wedge Rzy))\ \circ$
$\forall_L$
$R11 \leftrightarrow \neg\exists z(R1z \wedge Rz1)\ \circ$
$\leftrightarrow_L$
$R11, \neg\exists z(R1z \wedge Rz1))\ \circ$
$\circ\ R11, \neg\exists z(R1z \wedge Rz1)$
$\neg_L$
$\neg_R$
$\circ\ \exists z(R1z \wedge Rz1)$
$\exists z(R1z \wedge Rz1)\ \circ$
$\exists_R$
$\exists_L$
$\circ\ R11 \wedge R11$
$\forall y\,(R1y \leftrightarrow \neg\exists z(Ryz \wedge Rzy)), R12 \wedge R21\ \circ$
$\wedge_R$
$\wedge_L$
$\bullet\ R11\ \ \bullet\ R11$
$R12, R21\ \circ$
$\forall_L$
$R12 \leftrightarrow \neg\exists z(R2z \wedge Rz2)\ \circ$
$\leftrightarrow_L$
$R12, \neg\exists z(R2z \wedge Rz2)\ \circ$
$\bullet\ R12, \neg\exists z(R2z \wedge Rz2)$
$\neg_L$
$\circ\ \exists z(R2z \wedge Rz2)$
$\exists_R$
$\circ\ R21 \wedge R12, R22 \wedge R22$
$\wedge_R$
$\bullet\ R21\ \ \bullet\ R12$

Figure 8.4: A tableau which proves that the Quine-paradox is not satisfiable.

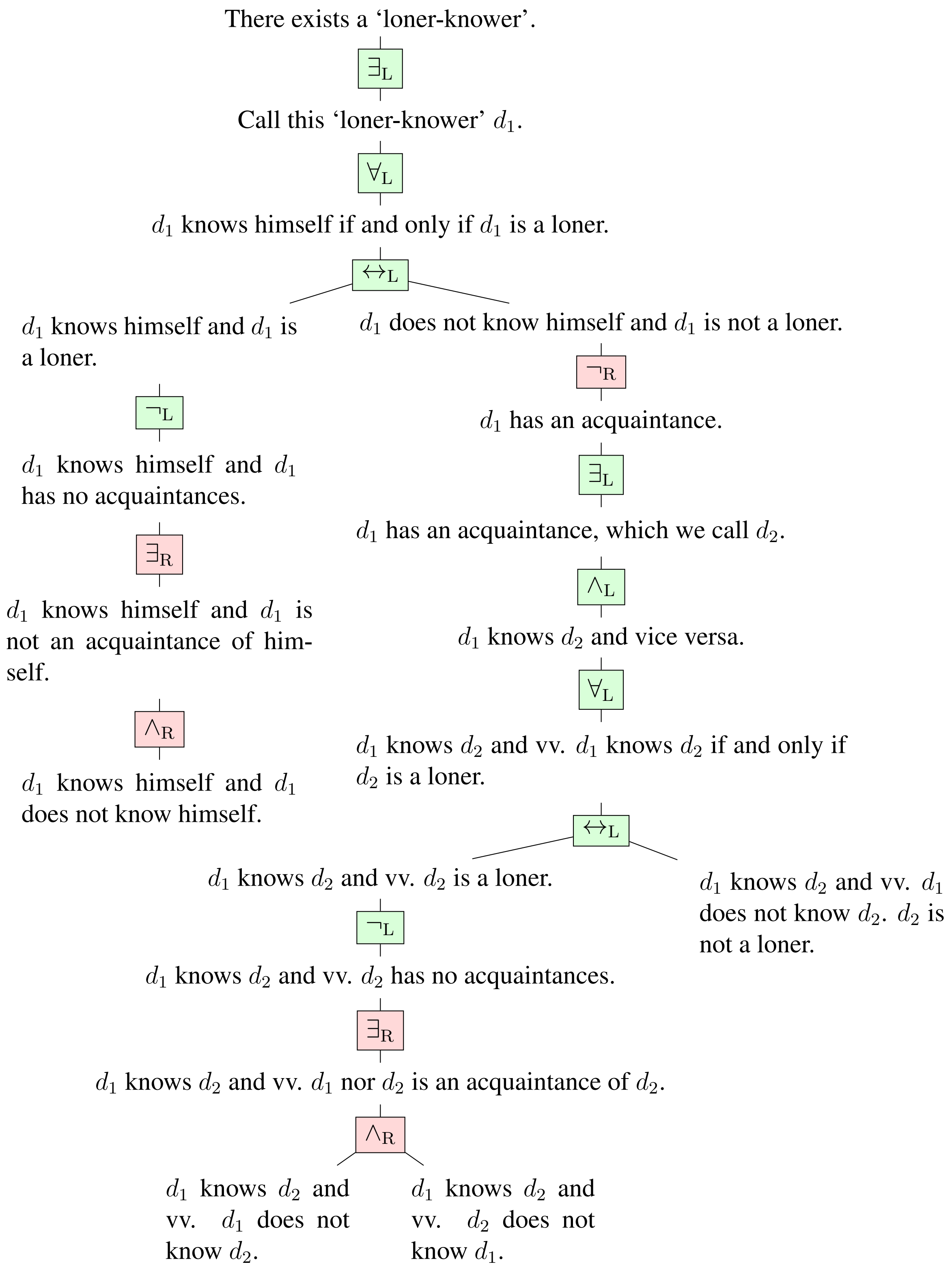
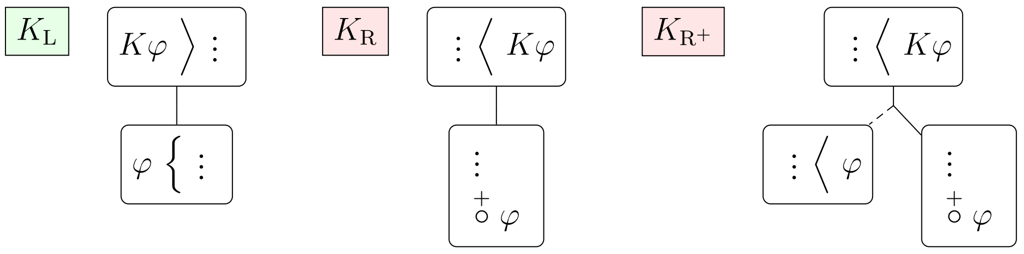

There exists a 'loner-knower'.
$\exists_{\mathrm{L}}$
Call this 'loner-knower' d_1.
$\forall_{\mathrm{L}}$
d_1 knows himself if and only if d_1 is a loner.
$\leftrightarrow_{\mathrm{L}}$
d_1 knows himself and d_1 is a loner.
$\neg_{\mathrm{L}}$
d_1 knows himself and d_1 has no acquaintances.
$\exists_{\mathrm{R}}$
d_1 knows himself and d_1 is not an acquaintance of himself.
$\wedge_{\mathrm{R}}$
d_1 knows himself and d_1 does not know himself.
d_1 does not know himself and d_1 is not a loner.
$\neg_{\mathrm{R}}$
d_1 has an acquaintance.
$\exists_{\mathrm{L}}$
d_1 has an acquaintance, which we call d_2.
$\wedge_{\mathrm{L}}$
d_1 knows d_2 and vice versa.
$\forall_{\mathrm{L}}$
d_1 knows d_2 and vv. d_1 knows d_2 if and only if d_2 is a loner.
$\leftrightarrow_{\mathrm{L}}$
d_1 knows d_2 and vv. d_2 is a loner.
$\neg_{\mathrm{L}}$
d_1 knows d_2 and vv. d_2 has no acquaintances.
$\exists_{\mathrm{R}}$
d_1 knows d_2 and vv. d_1 nor d_2 is an acquaintance of d_2.
$\wedge_{\mathrm{R}}$
d_1 knows d_2 and vv. d_1 does not know d_2.
d_1 knows d_2 and vv. d_2 does not know d_1.
d_1 knows d_2 and vv. d_1 does not know d_2. d_2 is not a loner.

Figure 8.5: A tableau which proves that the Quine-paradox is not satisfiable.



$K_{\rm L}$
$K\varphi \rangle \vdots$
$\varphi \{ \vdots$
$K_{\rm R}$
$\vdots \langle K\varphi$
$\overset{+}{\circ}\ \varphi$
$K_{\rm R+}$
$\vdots \langle K\varphi$
$\vdots \langle \varphi$
$\overset{+}{\circ}\ \varphi$

(8.34)

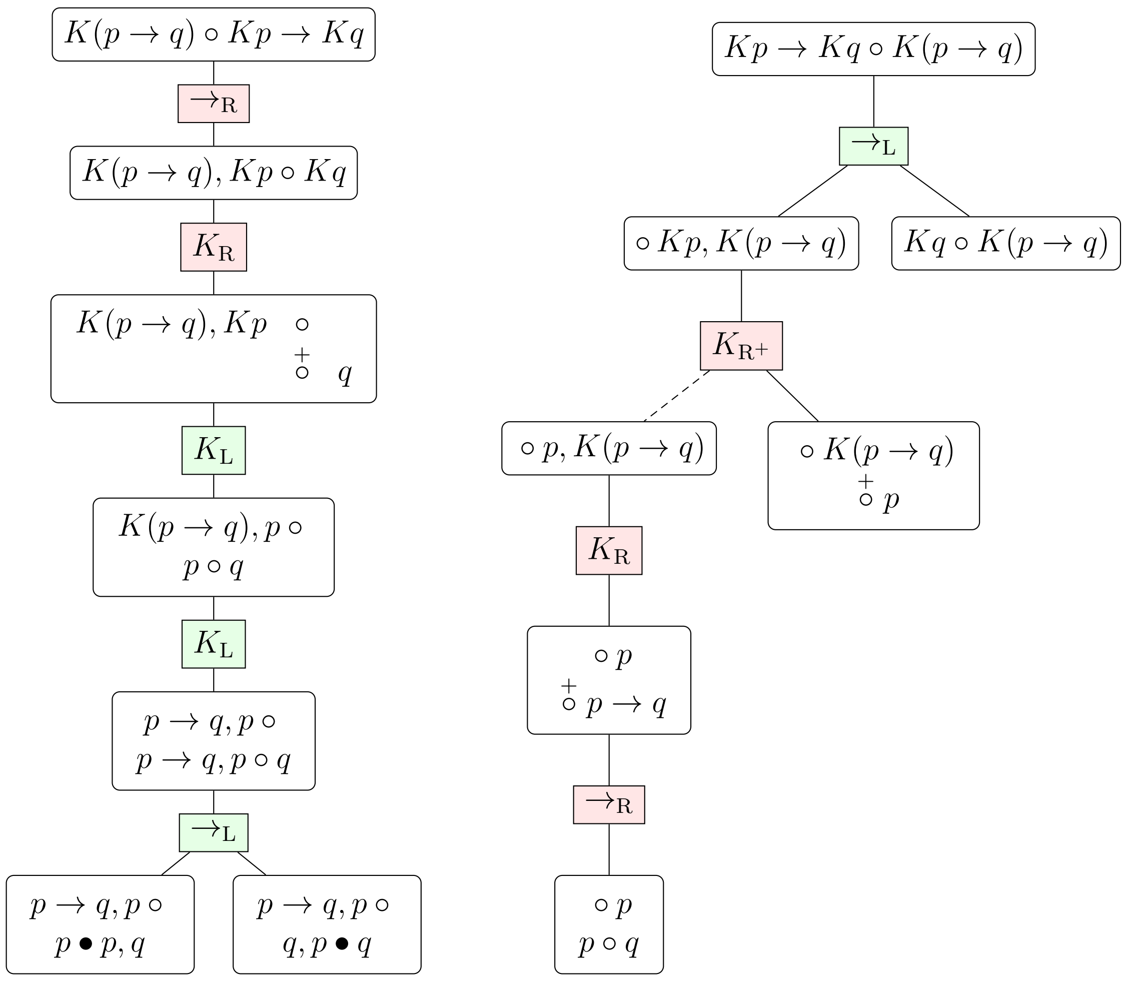
$K_{\rm L}$: If a formula $K\varphi$ appears on the left part of at least one of the sequents in the box then remove this formula from those sequents and add φ to all the sequents in the box.

$K_{\rm R}$: 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.

$K_{\rm R+}$: 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)

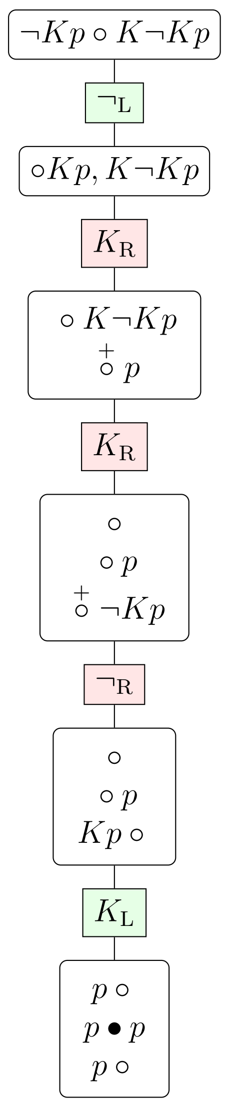
Before we may jump to conclusions we need to be precise about closed and open multi-sequents. A multi-sequent is *closed* if it contains an impossible sequent containing contradictory information, i.e., a formula which appears on the left and on the right part of the sequent. All the worlds as described 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)

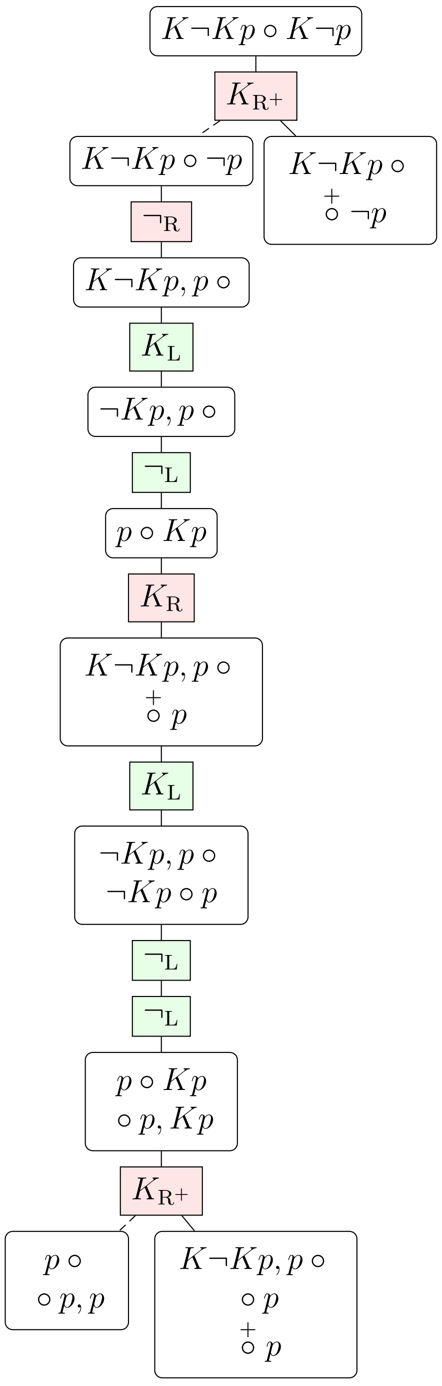
The tableau ends with a description of a single 'impossible' possible worlds model. In fact it tells us that a counter-model requires at least one impossible world at which p is both true and false, and therefore, a counter-model for negative introspection does not exist.

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)

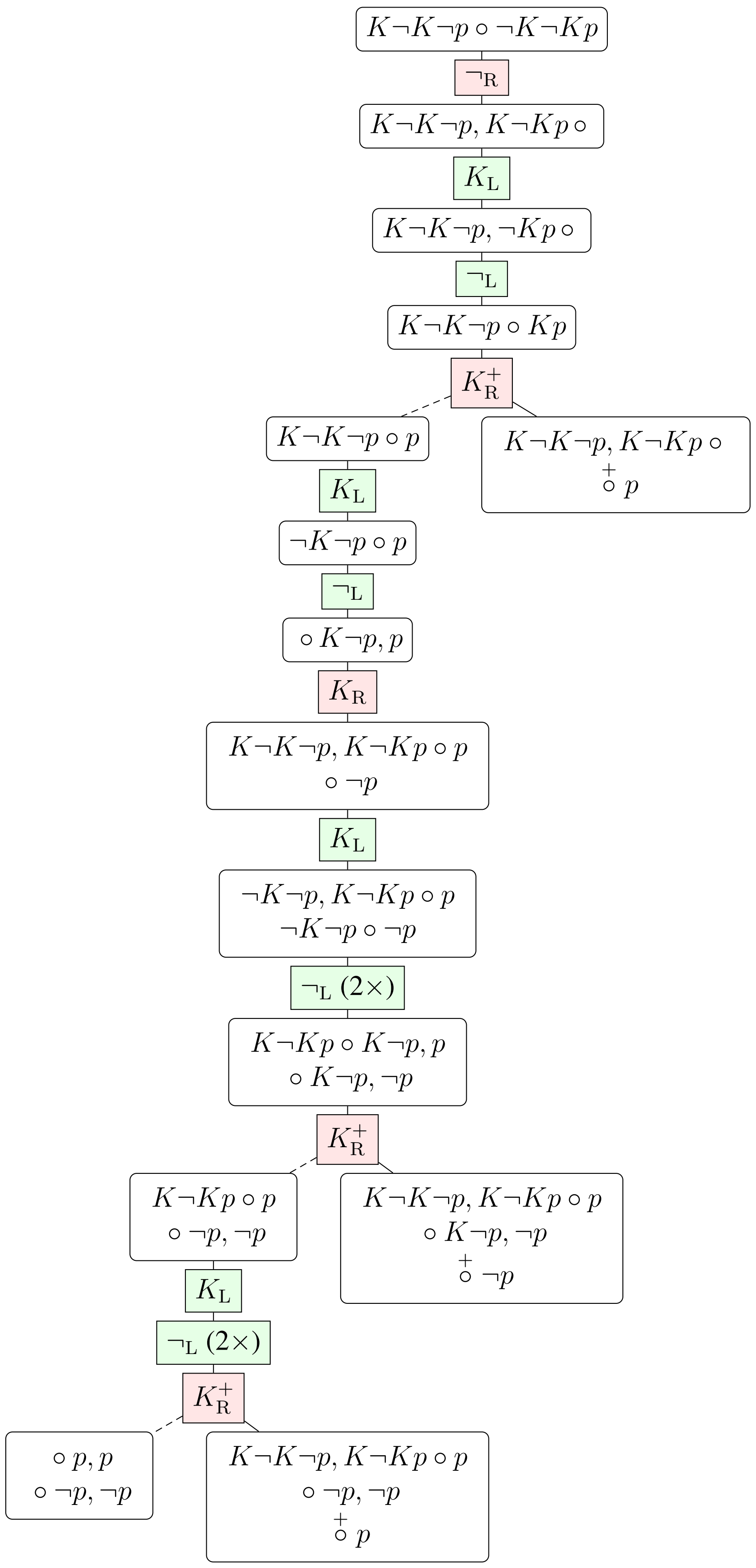
The counter-model which has been found in the left-most branch contains two worlds, one which verifies p and one which falsifies p. In both worlds the agent does not know that p and so $K\neg Kp$ is true in the first world (and also in the second), but $K\neg p$ is false in this world because p is false 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.


$K\neg K\neg p \circ \neg K \neg K p$
$\neg_R$
$K\neg K\neg p, K\neg K p \circ$
K_L
$K\neg K\neg p, \neg K p \circ$
$\neg_L$
$K\neg K\neg p \circ K p$
K_R^+
$K\neg K\neg p \circ p$
$K\neg K\neg p, K\neg K p \circ$ $\overset{+}{\circ}\, p$
K_L
$\neg K\neg p \circ p$
$\neg_L$
$\circ\, K\neg p, p$
K_R
$K\neg K\neg p, K\neg K p \circ p$ $\circ\, \neg p$
K_L
$\neg K\neg p, K\neg K p \circ p$ $\neg K \neg p \circ \neg p$
$\neg_L$ (2×)
$K\neg K p \circ K\neg p, p$ $\circ\, K\neg p, \neg p$
K_R^+
$K\neg K p \circ p$ $\circ\, \neg p, \neg p$
$K\neg K\neg p, K\neg K p \circ p$ $\circ\, K\neg p, \neg p$ $\overset{+}{\circ}\, \neg p$
K_L
$\neg_L$ (2×)
K_R^+
$\circ\, p, p$ $\circ\, \neg p, \neg p$
$K\neg K\neg p, K\neg K p \circ p$ $\circ\, \neg p, \neg p$ $\overset{+}{\circ}\, p$

(8.38)