1 Introduction: Search for a Counterexample

Remark 8.1.1 In this section we describe an elegant and algorithm which can replace the method of truth tables and which is naturally extended to first-order logic (which the method of truth tables from the previous section can not). It is called the *method of semantic tableaux*, although a more descriptive name might be *satisfiability trees*.

To test whether a proposition α with n propositional symbols is a tautology, we need to construct a truth table with 2^n entries. However, a single counterexample is sufficient to rule this possibility out. Perhaps, it would be more efficient to systematically search for a counterexample to α, that is, a valuation v in which $v(\alpha) = \mathbf{F}$. For the moment, we will keep track of whether a proposition β is true or false in our search by attaching a truth value to β: so, $\mathbf{T}\beta$ means β is to be true and $\mathbf{F}\beta$ is to be false. We will start the root of our counterexample tree with $\mathbf{F}\alpha$ and consider what we must do to build a valuation for which $v(\alpha) = \mathbf{F}$.

Example 8.1.2 Let $\alpha = (P \to Q) \to (\neg Q \to \neg P)$. Initially, we start with a one-node tree labeled:

$$\langle \mathbf{F}, (P \to Q) \to (\neg Q \to \neg P) \rangle.$$

In order to make α false, we must make $(P \to Q)$ true and $(\neg Q \to \neg P)$ false:

$$\begin{array}{c} \langle \mathbf{F}, (P \to Q) \to (\neg Q \to \neg P) \rangle \\ \langle \mathbf{T}, (P \to Q) \rangle \\ \langle \mathbf{F}, (\neg Q \to \neg P) \rangle \end{array}$$

To make $(P \to Q)$ true we can either make P false or make Q true. We will need to split the tree at this point to take care of each case:

$$\begin{array}{c} \langle \mathbf{F}, (P \to Q) \to (\neg Q \to \neg P) \rangle \\ \langle \mathbf{T}, (P \to Q) \rangle \\ \langle \mathbf{F}, (\neg Q \to \neg P) \rangle \\ \diagup \quad \diagdown \\ \langle \mathbf{F}, P \rangle \quad \langle \mathbf{T}, Q \rangle \end{array}$$

We continue on each leaf, starting with the left branch. To make $(\neg Q \to \neg P)$ false, we make $\neg Q$ be true and $\neg P$ false:

$$\begin{array}{c} \langle \mathbf{F}, (P \to Q) \to (\neg Q \to \neg P) \rangle \\ \langle \mathbf{T}, (P \to Q) \rangle \\ \langle \mathbf{F}, (\neg Q \to \neg P) \rangle \\ \diagup \quad \diagdown \\ \langle \mathbf{F}, P \rangle \quad \langle \mathbf{T}, Q \rangle \\ \langle \mathbf{T}, \neg Q \rangle \\ \langle \mathbf{F}, \neg P \rangle \end{array}$$

In order to make $\neg P$ false, we make P true:

$$\begin{array}{c}
\langle \mathbf{F}, (P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P) \rangle \\
\langle \mathbf{T}, (P \rightarrow Q) \rangle \\
\langle \mathbf{F}, (\neg Q \rightarrow \neg P) \rangle \\
\hline
\begin{array}{cc}
\langle \mathbf{F}, P \rangle & \langle \mathbf{T}, Q \rangle \\
\langle \mathbf{T}, \neg Q \rangle & \\
\langle \mathbf{F}, \neg P \rangle & \\
\langle \mathbf{T}, P \rangle & \\
\otimes &
\end{array}
\end{array}$$

The left branch requires that we make P true and P false, which is impossible. We say that this branch is *closed*. I have added $\otimes$ at the end of this branch to remind us that this branch is now closed. We still have the right branch to work on, so we will again try to make $(\neg Q \rightarrow \neg P)$ false by making $\neg Q$ true and $\neg A$ false on the right branch

$$\begin{array}{c}
\langle \mathbf{F}, (P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P) \rangle \\
\langle \mathbf{T}, (P \rightarrow Q) \rangle \\
\langle \mathbf{F}, (\neg Q \rightarrow \neg P) \rangle \\
\hline
\begin{array}{cc}
\langle \mathbf{F}, P \rangle & \langle \mathbf{T}, Q \rangle \\
\langle \mathbf{T}, \neg Q \rangle & \langle \mathbf{T}, \neg Q \rangle \\
\langle \mathbf{F}, \neg P \rangle & \langle \mathbf{F}, \neg P \rangle \\
\langle \mathbf{T}, P \rangle & \\
\otimes &
\end{array}
\end{array}$$

To make $\neg Q$ true we make Q false, so we will add this to the right branch:

$$\begin{array}{c}
\langle \mathbf{F}, (P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P) \rangle \\
\langle \mathbf{T}, (P \rightarrow Q) \rangle \\
\langle \mathbf{F}, (\neg Q \rightarrow \neg P) \rangle \\
\hline
\begin{array}{cc}
\langle \mathbf{F}, P \rangle & \langle \mathbf{T}, Q \rangle \\
\langle \mathbf{T}, \neg Q \rangle & \langle \mathbf{T}, \neg Q \rangle \\
\langle \mathbf{F}, \neg P \rangle & \langle \mathbf{F}, \neg P \rangle \\
\langle \mathbf{T}, P \rangle & \langle \mathbf{F}, Q \rangle \\
\otimes & \otimes
\end{array}
\end{array}$$

We see the right branch is impossible since it forces us to make Q both true and false. It is now closed. Since every branch is closed, we can conclude it is impossible that $(P \rightarrow Q) \rightarrow (\neg Q \rightarrow \neg P)$ can be false, so it must be a tautology.

2 Rules for constructing semantic tableaux

Remark 8.2.1 The use of the letters **T** and **F** was heuristically useful because it reminded us of our commitments. In what follows, we will replace a commitment $\mathbf{T}\alpha$ with α and a commitment $\mathbf{F}\alpha$ with $\neg\,\alpha$. The rules for constructing semantic tableaux utilize our unifying notation. A semantic tableaux is a binary branching tree (similar to formation trees). We start the tree with a proposition at the root that we want to find a satisfying assignment. The idea is that each branch should be though of as representing the conjunction of the propositions appearing on it, and the tree itself as representing a disjunction of the branches. We will have rules which allow us to extend a branch on the semantic tableaux. The rules are of two types: type-A rules and type-B rules, where these rules are based on the unifying notation from Definition 7.1.2 of Lecture 7.

A type-A proposition has two components, A_1 and A_2, which are *conjunctively related* to A in that

$$v(A) = v(A_1 \wedge A_2).$$

Each type A proposition will have a rule of the form

$$\begin{array}{c} A \\ | \\ A_1 \\ A_2 \end{array}$$

which allows us to add the components A_1 and A_2 to any branch on which A occurs. This captures the conjunctive behavior of A in that a commitment to A means a commitment to both A_1 and A_2.

A type-B proposition also has two components, B_1 and B_2, which are *disjunctively related* to B in that

$$v(B) = v(B_1 \vee B_2).$$

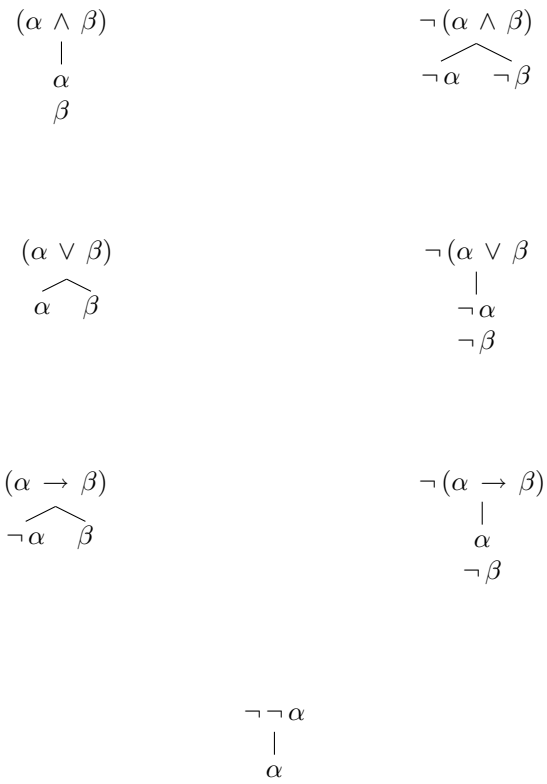
Each type B proposition will have a rule of the form

$$\begin{array}{ccc} & B & \\ & \diagup\ \diagdown & \\ B_1 & & B_2 \end{array}$$

which allows us to create two new branches on any path on which B occurs, and add the component B_1 to one branch and B_2 to the second branch. This captures the disjunctive behavior of B in that a commitment to B means a commitment to B_1 or B_2, but not necessary both.

Definition 8.2.2 (Rules for the construction of semantic tableaux) We explicitly state each rule following the discussion in the previous remark, and our unifying notation from Definition 7.1.2. Each connective (except $\neg$) has a rule for handling it and its negation. There are no rules for literals (propositional symbols, their negations, $\bot$ and $\top$).

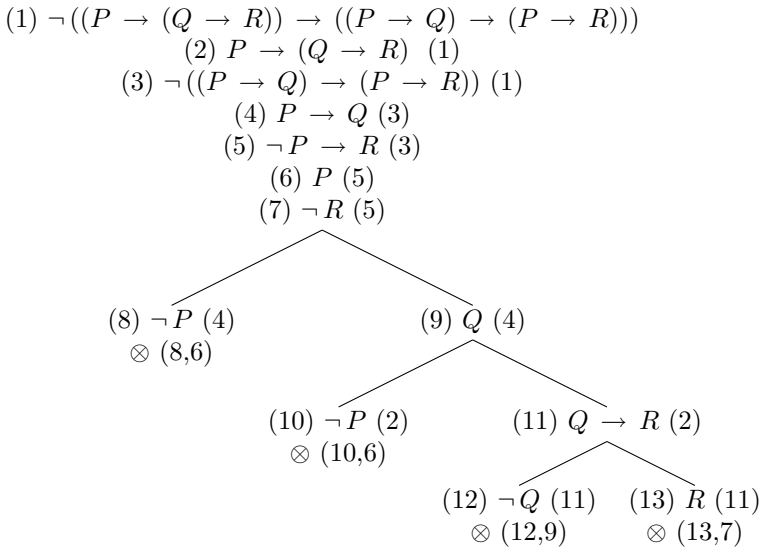
Before we provide a formal definition of a semantic tableaux, we will work through some examples of tableaux constructed according to the rules. If at any point in our construction of a branch we encounter a proposition β and its negation $\neg\,\beta$, or the atom $\bot$, we will close the branch off, since we have discovered that it is impossible to make all the propositions on the branch true.



Example 8.2.3 In this example we want to show that $((P \to (Q \to R)) \to ((P \to Q) \to (P \to R)))$ is a tautology. We will try to construct a counterexample by producing a valuation v which makes its negation true. So, we begin our tree with the root labeled

$$\neg((P \to (Q \to R)) \to ((P \to Q) \to (P \to R))).$$

From this point onward we may only extend our tree according to the rules in Definition 2.2.2. We also close off a branch when we discover a contradiction on that branch. I have numbered lines on the left and placed to the right the proposition from which it was derived. I have also given the contradictory lines. These are to show you how we have extended the tree, and are not formally part of the tableau.



Every path on this tableaux is contradictory, so our attempt to make the proposition at the root true has led to failure. This means that the proposition

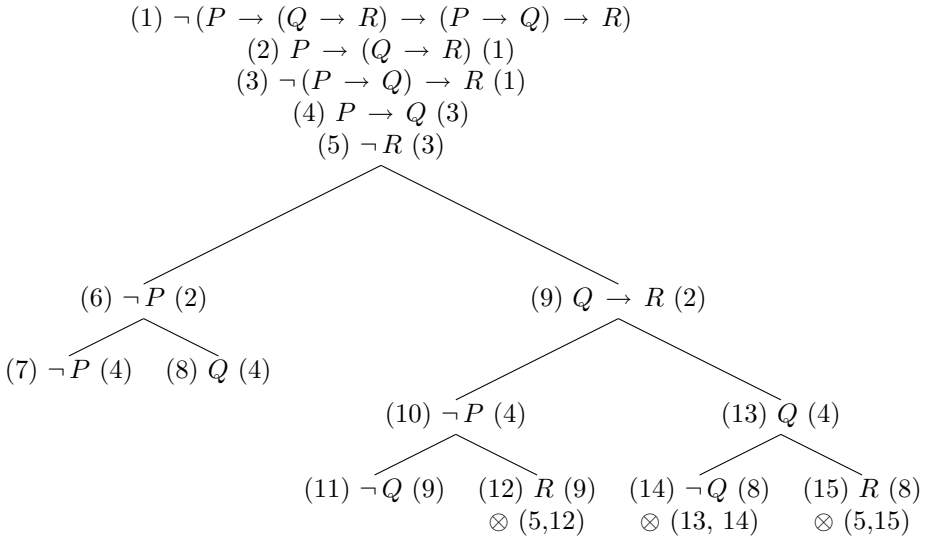
$$(P \to (B \to R)) \to ((P \to B) \to (P \to R)).$$

is a tautology. We will take this tree, constructed according to the rules of semantic tableaux, as a *proof* that it is a tautology.

Example 8.2.4 In this example we will try to construct a proof that the proposition $(P \to (Q \to R) \to (P \to Q) \to R)$ is a tautology. It is not, and you will see that the final tableaux will have open branches from which we can construct falsifying assignments. We begin the tableau by trying to refute this proposition:

$$\neg(P \to (Q \to R) \to (P \to Q) \to R).$$

Here is a tableau:



There are three open branches that still have no contradictions. Notice that any further application of a rule on any of these paths will simply extend to a proposition *already on the path*. A tableau in which every branch is either contradictory, or in which any further extension by the tableaux rules only adds redundant propositions, is said to be *finished*. Any branches in a finished tree without contradictions are said to be

open. A finished tree on which every branch contains a contradiction is a *contradictory* tree, and constitutes a proof that it is impossible to make the proposition at the root true.

Although, we have failed in our attempt to construct a tableau proof of the proposition $(P \to (Q \to R) \to (P \to Q) \to R)$, we can extract an assignment $\mathcal{A}$ to the propositional symbols P, Q, R along any noncontradictory path in which in which valuation from any non-contradictory path that makes the proposition

$$v_{\mathcal{A}}\big((P \to (Q \to R) \to (P \to Q) \to R)\big) = \mathbf{F}.$$

For example, the leftmost path is noncontradictory and is committed to making both P and R false on lines (5) and (6). It makes no commitment to the truth value of Q since neither Q nor $\neg Q$ lies on the path; so it is enough that the assignment $\mathcal{A}$ makes $\mathcal{A}(P) = \mathbf{T}$ and $\mathcal{A}(R) = \mathbf{F}$ and may give Q any truth value. The following partial truth table verifies the counterexamples we have extracted from this path.

P	Q	R	$P \to (Q \to R)$	$P \to Q$	$(P \to Q) \to R$
F	**T**	**F**	**T**	**T**	**F**
F	**F**	**F**	**T**	**T**	**F**

The other paths lead to the same assignments. Eventually we will prove that a counterexample can always be extracted from a non-contradictory path in a finished tableau.

3 The Tableau Method Formalized.

Remark 8.3.1 We now make precise the ideas introduced in the last section. The definition of tableaux provided next describes all possible tableaux. These include *finite tableaux* defined inductively, and infinite tableaux which are the limit of a sequence of finite tableaux. We could get by with only finite tableaux for now, but we will have to have infinite tableaux when we extend the tableau method to first-order logic.

Definition 8.3.2 A *finite tableau* is a binary tree, each node labeled with a proposition, and which satisfies the following inductive definition:

(a) All one-node trees labeled with a proposition are finite tableaux.

(b) If τ is a finite tableau, π a path through τ, and A on π, then the extension placing the components A_1 and the A_2 on π is also a finite tableau.

(c) If τ is a finite tableau, π a path through τ, and $\mathcal{B}$ on π, then the extension of π placing the component B_1 on the left branch and the B_2 on the right branch is also a finite tableau.

If $\tau_0, \tau_1, \ldots, \tau_n, \ldots$ is a (finite or infinite) sequence of finite tableaux such that for each $n \geq 0$, τ_{n+1} is constructed from τ_n by an application of either (b) or (c), then $\tau = \cup_n \tau$ is a *tableau*.

Definition 8.3.3 (contradictory, finished tableau) Let τ be a tableau and π a path through τ.

- A proposition α on π has been *reduced on* π if one of the following three conditions apply:
  - (i) α is a literal.
  - (ii) $\alpha = A$ and both its A_1 and A_2 components are on π.
  - (iii) $\alpha = B$ and at least one of its B_1 or B_2 component are on π.
- π is *contradictory* if a proposition and its negation are on π. π is *finished* if it is contradictory or every proposition on π has been reduced on π.
- τ is *finished* if every path through τ is finished.
- τ is *contradictory* if every path through τ is contradictory.

Definition 8.3.4 (tableau proof) A *tableau proof* of a proposition α is a contradictory tableau whose root is labeled $\neg\,\alpha$. A proposition α is *tableau provable*, denoted by $\vdash \alpha$, if α has a tableau proof.

We now show that we we only needed finite tableaux for proving or refuting that propositions are tautologies.

Theorem 8.3.5 There is a finished finite tableau τ for each possible root entry α.

Proof. The proof is by induction (using the unifying notation) on the proposition α.

(atomic). If α is a literal, then the tableau consisting of α is finished.

(type-A). If $X = A$, then the the following is a tableau:

$$\begin{array}{c} A \\ | \\ A_1 \\ A_2 \end{array}$$

By the induction hypothesis A_1 has a finished finite tableau τ_1 and A_2 has a finished finite tableau τ_2. First, remove the root from both τ_1 (the root is A_1) and τ_2 (the root is A_2). Append τ_1 after A_2 in the above tableau, so this is still a finite tableau whose only unfinished node is the occurrence of A_2 above. On each open branch append a copy of τ_2. This is still a tableau since τ_2 is placed only on branches which contain A_2. Call this new tableau τ, which is a finite finished tableau (since the occurrence of A_2 above is now finished).

(type-B). If $X = B$, then the the following is a tableau:

$$\begin{array}{ccc} & B & \\ & \diagup\ \diagdown & \\ B_1 & & B_2 \end{array}$$

By the induction hypothesis B_1 has a finished finite tableau τ_1 and B_2 has a finished finite tableau τ_2. Let τ be the tableau obtained by appending τ_1 below B_1 and τ_2 below B_2, deleting the redundant copy of B_1 and B_2. Then τ is a finite finished tableau.

So, every proposition has a finished tableau where it is labeled at the root. □

Remark 8.3.6 The proof of the last theorem does give an algorithm for constructing finished tableau for propositions, but the algorithm is not very efficient. Its only use is to establish that finished tableaux are finite.

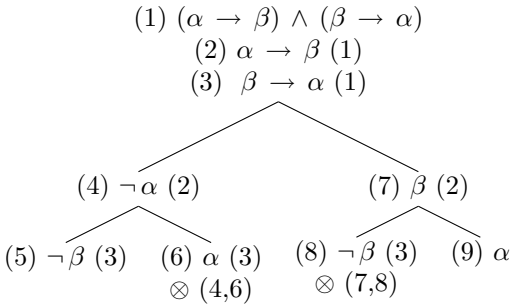
4 Tableaux Rules for the Biconditional

Remark 8.4.1 We did not provide tableaux rules for the biconditional, as we now take the biconditional to be defined by:

$$(\alpha \leftrightarrow \beta) =_{\text{def}} (\alpha \rightarrow \beta) \wedge (\beta \rightarrow \alpha).$$

Since $(\alpha \leftrightarrow \beta)$ is true if both β and α have the same truth value, and is false if α and β have different truth values, you can verify the correct tableau rules for the biconditional:

We can derive the correct tableaux rules for the biconditional using the tableaux rules for conjunction and the conditional and the definition of the biconditional:



One branch has $\neg\alpha$ and $\neg\beta$, and the other open branch contains α and β. But this is just what our rule says we should do. The rule for $\neg(\alpha \to \beta) \wedge (\beta \to \alpha)$ is also easily derived from the rules for conditionals and conjunctions.

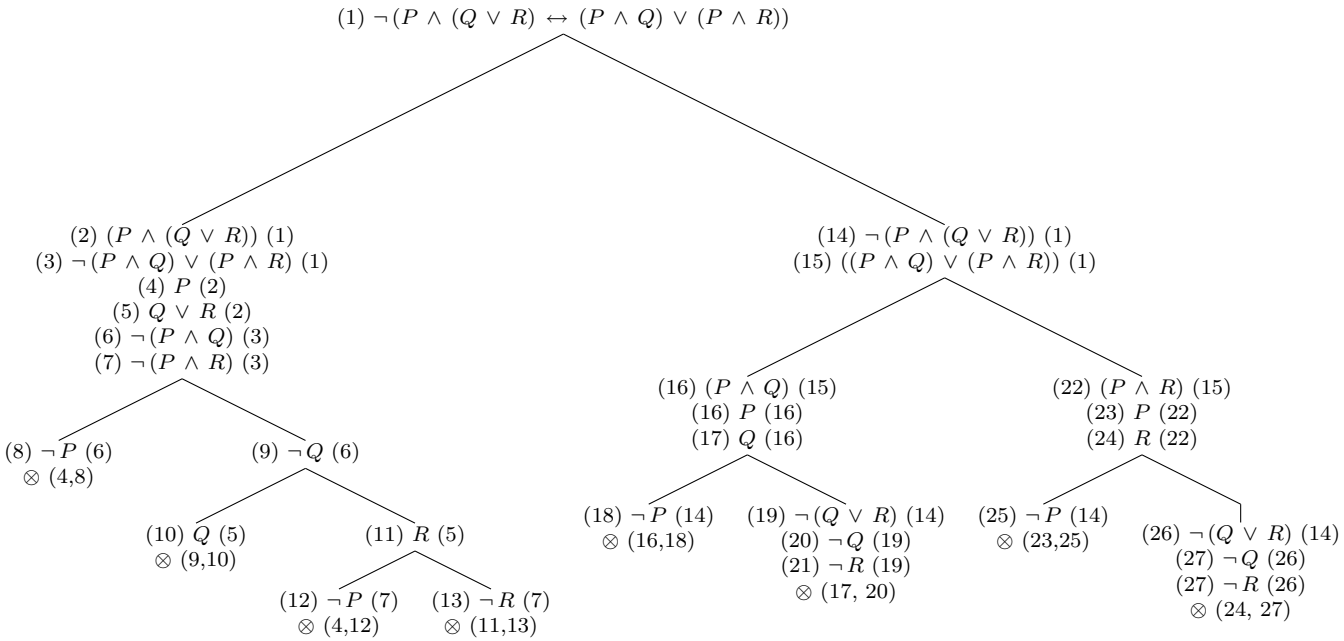
Example 8.4.2 (testing logical equivalence) We can to test for tautological equivalence, $\simeq$, by testing that a biconditional is a tautology. This is exactly as we showed in Lecture 5. For example, to test whether

$$(P \wedge (Q \vee R) \simeq (P \wedge Q) \vee (P \wedge R)).$$

try to refute **??**:

$$\neg(P \wedge (Q \vee R) \leftrightarrow (P \wedge Q) \vee (P \wedge R)).$$

Apply the extended rules for biconditional in remark 2.4.1.



All paths through this tableau are contradictory, so the tableau is finished. You can verify by the truth table method that

$$P \wedge (Q \vee R) \leftrightarrow (P \wedge Q) \vee (P \wedge R).$$

is a tautology. This tableau is a proof that it is.

5 Tableaux Rules for Additional Connectives

Remark 8.5.1 We can extend the tableaux rules to allow $\uparrow$ (NAND) and $\downarrow$ (NOR) from Lecture 6. We extended the unifying notation in Remark 7.2.4 of Lecture 7, and based on this we add the following rules:

$$\begin{array}{c} (\alpha \uparrow \beta) \\ \diagup \quad \diagdown \\ \neg\alpha \quad \neg\beta \end{array} \qquad \begin{array}{c} \neg(\alpha \uparrow \beta) \\ | \\ \alpha \\ \beta \end{array}$$

$$\begin{array}{c} (\alpha \downarrow \beta) \\ | \\ \neg\alpha \\ \neg\beta \end{array} \qquad \begin{array}{c} \neg(\alpha \downarrow \beta) \\ \diagup \quad \diagdown \\ \alpha \quad \beta \end{array}$$

Example 8.5.2 Here is an example which shows that $(P \downarrow P) \downarrow P$ is a tautology

$$\begin{array}{c} (1)\ (P \downarrow P) \downarrow P \\ (2)\ \neg(P \downarrow P)\ (1) \\ (3)\ \neg P\ (1) \\ \diagup \quad \diagdown \\ (4)\ P\ (2) \quad (5)\ P\ (2) \\ \otimes(3),(4) \quad \otimes(3),(5) \end{array}$$

Example 8.5.3 Here is an example which shows that $((P \uparrow (Q \vee R)) \wedge (Q \vee R)) \rightarrow \neg P$ is a tautology.

$$\begin{array}{c} (1)\ \neg((P \uparrow (Q \vee R)) \wedge (Q \vee R)) \rightarrow \neg P \\ (2)\ (P \uparrow (Q \vee R)) \wedge (Q \vee R)\ (1) \\ (3)\ \neg\neg P\ (1) \\ (4)\ P\ (3) \\ (5)\ (P \uparrow (Q \vee R))\ (2)\ (6)\ (Q \vee R)\ (2) \\ \diagup \quad \diagdown \\ (7)\ \neg P\ (2) \quad (8)\ \neg(Q \vee R)\ (2) \\ \otimes(4),(7) \quad \otimes(6),(8) \end{array}$$

6 Soundness and Completeness

Remark 8.6.1 We have a method of proof for tautologies by constructing truth tables and a method by constructing theorems of semantic tableaux. Do these two methods agree? Does every tautology α have a contradictory semantic tableau whose root is $\neg\alpha$? If we do produce a contradictory semantic tableau for $\neg\alpha$ can we conclude it is a tautology? Perhaps we missed some assignment?

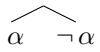
A proof procedure is *sound* if it proves only tautologies and *complete* if it proves all tautologies. The method of truth tables is obviously correct and complete by Lemma 4.1.4 of Lecture 4, the truth of a proposition depends only on its support. It is straightforward to see that the method of tableaux is sound, given our interpretation that each branch is a conjunction of our commitments and the tableau itself represents

a disjunction over all branch-commitments. The type-A rules replace α with its components α_1 and α_2 on the same branch, and this is justified by the fact that $\alpha \simeq \alpha_1 \wedge \alpha_2$. Similarly the type-$B$ rules replace β with a branching tree whose left branch contains the component β_1 and whose right branch contains the component β_2. This is justified by the fact that $\beta \simeq \beta_1 \wedge \beta_2$.

The key question is whether we have a tableau proof for every tautology. Our rules were guided by the semantics, so we expect them to be sound, but have we really captured everything necessary for establishing a tautology? If α is provable by a semantic tableau and $\alpha \rightarrow \beta$ is also provable, does it follow that β is provable? It is not like we can somehow *use* the contradictory tableaux for α and $\alpha \rightarrow \beta$ in our construction of a contradictory tableau for β. It is not at all apparent that there should be a contradictory tableau for β. It is certainly true that if α is a tautology and $\alpha \rightarrow \beta$ is, then β is.

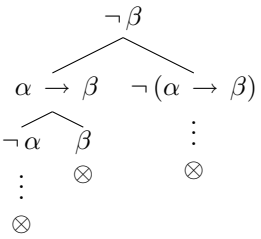
We can remedy this situation by introducing a new rule of inference.

Definition 8.6.2 Synthetic Tableaux A *synthetic tableau* uses all the rules for constructing semantic tableaux plus the following rule for any proposition:



This is known as the *splitting rule* and tableaux constructed using the splitting rule are called *synthetic tableaux*. Tableaux constructed without the splitting rule are called *analytic tableaux*.

Remark 8.6.3 The splitting rule is justified by the fact that $\alpha \vee \neg\alpha$ is a tautology, so one branch of the splitting rule must be true. Analytic tableaux only allow the components of a proposition to enter on a path; synthetic tableaux allow us to bring any proposition and its negation into the tableau. The strength of this is that now, if we have a tableau proof of α and $\alpha \rightarrow \beta$, we can produce a synthetic tableau proof of β. Here is how:



where we put the contradictory tableau for $\neg\alpha$ and $\neg(\alpha \rightarrow \beta)$ under these nodes in the tree.

In fact, adding the splitting rule does not increase the number of theorems we can prove, but it does allow us to use previous proofs. The reason is not at all obvious and is a consequence of the *completeness theorem* for the method of analytic tableaux. We will take this issue up again in a few lectures.