1 Finished Tableaux

Remark 26.1.1 We turn now to the problem of determining when a tableau is finished.

Type-A, type-B and type-D rules need only be applied once, since any further application adds no new information to the path. Once we have applied a type-D rule to a line, say from $(\exists x)P(x)$ to infer $P(a)$ for some parameter a new to the path, the original sentence contains no more information then the new sentence $P(a)$.

On the other hand, a type-C application is quite different. Suppose we have $(\forall x)P(x)$ on a line, and we add $P(t)$ for some variable-free term. This is does not yet exhaust the information found in the original sentence. It merely gives one instance of the universal fact asserted by $(\forall x)P(x)$.

Our goal, as in the propositional case, is to describe a systematic procedure to produce a tableau proof from Σ of a given sentence α. The Completeness Theorem for Semantic Tableaux will show that this procedure always succeeds in producing a proof, if α is indeed a logical consequence of Σ.

Remark 26.1.2 Let $\mathcal{L}$ be a first-order language and $A = \{a_0, a_1, a_2, \ldots\}$ be the set of parameters we use in constructing the domain for a tableaux. The *ground terms* of $\mathcal{L}^A$ are all terms in the extended language $\mathcal{L} \cup A$ which do not contain variables. (For now these are simply the constant symbols from $\mathcal{L}$ and the parameters from A.) Let $t_1, \ldots, t_n, \ldots$ be a list of all ground terms of our language $\mathcal{L}^A$ in the discussion that follows.

Definition 26.1.3 (finished tableau) Let $\tau = \cup_n \tau_n$ be a tableau for Σ, π a path through τ and ϕ is a signed formulae occurring on π.

- A node with label ϕ is *reduced* on π if either
  1. ϕ is type-A, type-B or type-D and for some j, τ_j came from τ_{j+1} by adding the components of ϕ to π.
  2. $\phi = \gamma$ is type-C and this is the i^{th} (for some i) occurrence of γ on π and there is a component $\gamma(t_i)$ and another occurrence of γ on nodes farther down on the path π.
- τ is *finished* if every occurrence of every node on τ is reduced on every noncontradictory path containing it and for each premiss ϕ from Σ, ϕ appears on every noncontradictory path.

Remark 26.1.4 The idea behind our treatment of type-C formulae such as $(\forall x)\phi(x)$ is that each instance of a ground term is eventually instantiated as $\phi(t_i)$ on every noncontradictory path. We can now show that there is a finished tableau for Σ for any initial sentence at its root by systematically constructing a finished tableau for Σ. The proof of the next theorem does produce an algorithm for constructing a finished tableau, but it is not very useful in practice, and in fact will always produce an infinite tableau.

Theorem 26.1.5 Let Σ be a set of sentences from a language $\mathcal{L}$. Then there exists a finished tableau for Σ.

Proof. Let $A = \{a_0, a_1, \ldots\}$ be the parameters used in constructing the tableau, $t_1, \ldots, t_n, \ldots$ be a list of all ground terms of the extended language $\mathcal{L}^A$, and $\phi_0, \phi_1, \ldots, \phi_n, \ldots$ be a list of all sentences in Σ. We will also need to keep track of the sentences from $\mathcal{L}^A$ that we put onto the tableau so that we can be sure we will eventually reduce them. We will call this list of sentences (together with the node on which it occurs) S.

The construction is by recursion on stages n. Each stage $n+1$ will take the finite tableau τ_n given at stage n and extend it to a finite tableau τ_{n+1} by applying one of the rules for constructing tableau. Each new node added to τ_n together with the node on which it occurs will be put onto the list S.

At stage $n = 0$, let τ_0 be the one-node tableau labeled with ϕ_0 and S contain just ϕ_0. Suppose at stage n we have constructed a finite tableau τ_n and the list $S = \beta_0, \beta_1, \ldots, \beta_m$ contains all sentences occurring in τ_n (together with the node on which it occurs). What we do at stage $n + 1$ will depend on whether it is an even or odd stage. We have to take steps to both reduce the nodes on τ as well as make sure that every sentence of Σ is eventually placed on each noncontradictory path of our tableau.

If $n = 2k$ is an even stage, then place ϕ_k on every noncontradictory branch of τ_n, and add each occurrence of ϕ_k to the list S. Since τ_n is finite, only finitely many nodes will be added to τ_n. Let the new tableau be τ_{n+1}.

If $n = 2k+1$ is an odd stage, then let β_i be the first unreduced node on the list S. We extend τ_n to τ_{n+1} according to the type of sentence that β_i is. If β_i is type-A or type-B then extend each path π containing β_i by placing the components of β_i onto the path in accordance with the appropriate rule. If $\beta_i = \delta$ is type-D, then extend each noncontradictory path π containing β_i by placing $\delta(a_j)$ on the path, where a_j is a parameter new to the tableau. If $\beta_i = \gamma$ is type-C and there are $j - 1$ predecessors of this node labeled with γ, then extend each noncontradictory path containing β_i by placing $\gamma(t_j)$ and γ on the path. We also extend the list S by placing all new nodes and their sentences onto the list. In each case, we have added only finitely many nodes to τ_n in constructing τ_{n+1}.

Let $\tau = \cup_n \tau_n$. Each τ_n is a finite tableau for Σ, so τ is a tableau for Σ. τ is a finished tableau. Let π be any noncontradictory path in τ. At stage $2k$ the sentence ϕ_k was placed on π, so π contains all sentences of Σ. If β is on a node on π then it is on the list S, so $\beta = \beta_i$ for some i, and at some odd stage $n = 2k + 1$ we will reduce β by placing its components onto π. Notice that if $\beta = \gamma$ is a type-C sentence, then $\gamma(t_j)$ will eventually be placed onto π.

So, τ is finished.

□

Remark 26.1.6 The definition of a finished tableau 1.0.2 is excessive for languages with no functions symbols. We could allow that a tableau τ for Σ is finished if all type-A, type-B and type-D nodes are reduced on each path through τ, ϕ is on every path through τ for each sentence $\phi \in \Sigma$ and for each type-C sentence γ on τ, each path π through τ containing γ and for each term t consisting of either a constant from $\mathcal{L}$ or a parameter appearing in τ, $\gamma(t)$ lies on π. (We also need to add that τ does contain some parameter, since the domain must be nonempty.)

Example 26.1.7 Consider the following unfinished tableau:

$$
\begin{array}{c}
(1)\ (\forall x)(\exists y)R(x,y) \\
(2)\ (\exists y)R(a_0,y)\ (1) \\
(3)\ (\forall x)(\exists y)R(x,y)\ (1) \\
(4)\ R(a_0,a_1)\ \ (2) \\
(5)\ (\exists y)R(a_1,y)\ (3) \\
(6)\ (\forall x)(\exists y)R(x,y)\ (3) \\
(7)\ R(a_1,a_2)\ (5) \\
\vdots
\end{array}
$$

This tableau will never be finished, even given the broader conditions of the previous remark. The structure we produce is $A = \{a_0, a_1, \ldots\}$ and $R^{\mathcal{A}} = \{\langle a_i, a_{i+1}\rangle : i \in \mathbb{N}\}$.

If we allowed our liberalized version of type-D from example 25.2.8 we would have avoided introducing the new parameter in line (4). The sentence on line (1) is true in the simple structure $\mathcal{A}$ with domain $A = \{a\}$ and $R^{\mathcal{A}} = \{\langle a, a\rangle\}$. However, there are noncontradictory tableau for single sentences which require infinite domains.