16.1 First-Order Tableaux

Since the evaluation of quantified formulas usually requires the evaluation of the formula for all possible elements of the universe, truth tables are unsuited for proving first-order formulas correct. Universes are usually infinite and even in a finite universe, the search space would quickly explode. The extension of the tableaux method to first-order logic, on the other hand, is quite straightforward. Let us consider an example.

$$\begin{array}{c}
F\,(\forall x)(Px \supset Qx) \supset ((\forall x)\ Px \supset (\forall x)\ Qx) \\
| \\
T\,(\forall x)(Px \supset Qx) \\
| \\
F\,(\forall x)\ Px \supset (\forall x)\ Qx \\
| \\
T\,(\forall x)\ Px \\
| \\
F\,(\forall x)\ Qx
\end{array}$$

Up to this point we have proceeded as in propositional logic. Now we have to startdecomposing quantifiers. The formula $(\forall x)Qx$ is false if Qx can be made false for at least one element k of the universe. Since the elements of the universe do not belong to the syntax of the formulas, we substitute x by a parameter a instead.

In the following step we decompose $T\,(\forall x)Px$. We know that $(\forall x)Px$ is true if Px is true for all elements of the universe. This means we can substitute any parameter for x and we choose a again, since this is useful for completing the proof. The remaining proof is straightforward and we get

$$\begin{array}{ccc}
 & F\,(\forall x)(Px \supset Qx) \supset ((\forall x)\ Px \supset (\forall x)\ Qx) & \\
 & | & \\
 & T\,(\forall x)(Px \supset Qx) & \\
 & | & \\
 & F\,(\forall x)\ Px \supset (\forall x)\ Qx & \\
 & | & \\
 & T\,(\forall x)\ Px & \\
 & | & \\
 & F\,(\forall x)\ Qx & \\
 & | & \\
 & F\,Qa & \\
 & | & \\
 & T\,Pa & \\
 & | & \\
 & T\,Pa \supset Qa & \\
 \swarrow & & \searrow \\
F\,Pa & & T\,Qa \\
| & & | \\
\times & & \times
\end{array}$$

Q: *Why did we decompose $F\,(\forall x)Qx$ before $T\,(\forall x)Px$ in the proof?*

The parameter a that we substituted for x was supposed to indicate that Qx can be made false by some yet unknown element of the universe. Since we do not know this element, a should be a *new parameter* – this way we make sure that we don't make any further assumptions about a by accidentally linking it to a parameter that was introduced earlier in the proof.

If we were to decompose T(∀x)Px *before* F(∀x)Qx then we would not be able to use a as parameter for Q, since it has already been used for P and is not unknown anymore. If we decompose F(∀x)Qx first, then a is still new. Choosing the same a for P is a decision we make afterwards.

In informal mathematics, quantifiers are handled in exactly the same way. When proving $(\forall x)(Px \wedge Qx) \supset (\forall x)Qx$ we assume $(\forall x)(Px \wedge Qx)$ and then try to show $(\forall x)Qx$. For this purpose we assume a to be arbitrary, but fixed, and try to prove Qa. Since we know $(\forall x)(Px \wedge Qx)$, we also know that $Pa \wedge Qa$ holds for the arbitrary a that we just chose and conclude that Qa is in fact the case. Note that it was crucial to have the a before instantiating $(\forall x)(Px \wedge Qx)$.

16.2 Extension of the unified notation

The above example shows that there are two different ways to handle quantifiers in tableaux proofs.

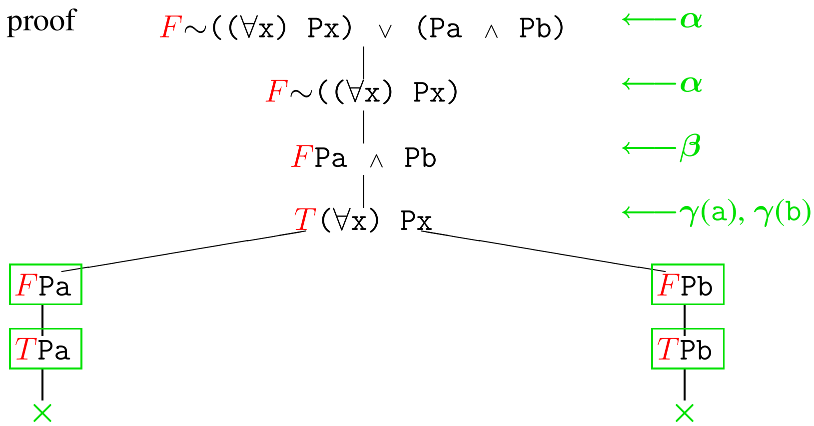
In the first case, we have formulas of the form T(∀x)A and, by duality, F(∃x)A, which we call formulas of type γ of *universal type*. γ-formulas are decomposed into TA[a/x] (and FA[a/x], respectively), where a is an arbitrary parameter. These formulas are often denoted by $\gamma(a)$.

In the other case, we have formulas of the form F(∀x)A and, by duality, T(∃x)A, which we call formulas of type δ of *existential type*. δ-formulas are decomposed into FA[a/x] (and TA[a/x], respectively), where a is a new parameter. These formulas are often denoted by $\delta(a)$ and the requirement that a must be new is usually called the *proviso* of the rule.

Altogether we have now four types of inference rules.[1]

α	β	γ	δ
α_1 α_2	$\beta_1 \mid \beta_2$	$\gamma(a)$ a arbitrary parameter	$\delta(a)$ a new parameter

Here is another example proof



[1]In calculi that use terms instead of parameters, the γ-rule allows a to be an arbitrary term (representing some object) whereas in the δ rule a must be a new variable, representing the fact that the element of the universe is unknow.

Note that in this proof, the γ-formula T(∀x) Px had to be instantiated twice to complete the proof. In general, formulas of universal type may be used arbitrarily often in a proof and therefore validity in first-order logic is not decidable.[2]

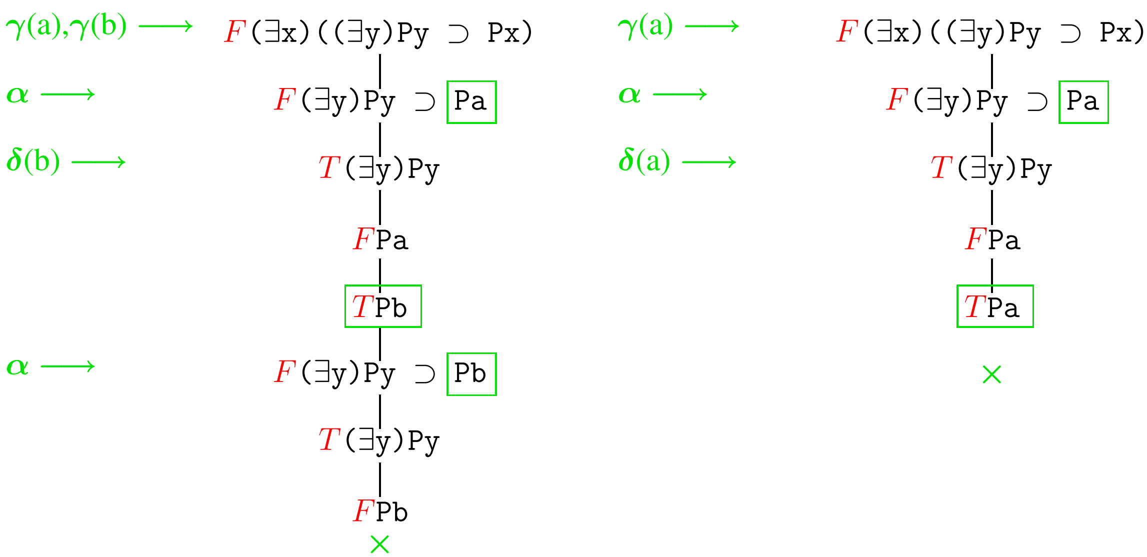
16.3 A liberalized δ rule

The proviso of the δ rule, which requires a to be a new parameter, is quite restrictive and makes formal proofs more complicated than they have to be. Actually, the proviso is more restrictive than it has to be. It is possible to liberalize the δ rule by replacing it by the following requirement:

> provided a is a new parameter
> or a was not previously introduced on the same path by a δ rule, does not occur in δ, and no parameter in δ was previously generated by a δ rule

In other words, if a does already occur in the proof then we may use it in a δ rule if it was generated by some γ rule. The rationale is that this γ rule could also be applied later and use the parameter a at that point ... after the δ rule has introduced it. Thus the fact that the γ rule appears earlier in the proof should not affect the parameters that the δ rule is permitted to use.

The following example shows the advantages of using a liberalized δ rule. In the proof on the left, the (original) the δ rule, which can only be applied after the first application of the γ rule, cannot use the parameter a because it already occurs in the proof. It has to use a new parameter b instead and we have to apply the γ rule again to get the formula FPb. Using the liberalized δ rule instead makes the proof on the right much shorter.



[2]This argument only appeals to the intuition. The actual proof of the undecidability of first-order logic is more complex, since one has to show that there is no other way to determine that a formula is not valid.

16.4 First-Order Gentzen Systems

Like analytic tableaux, Gentzen systems can easily be adapted to first-order logic. In propositional logic Gentzen systems were isomorphic to block tableaux, which in turn were just a different way of writing down tableau proofs – keeping track of a set S of all formulas in the tree that still could be decomposed yet unused form. Thus we get the proof rules of first-order Gentzen systems for sequents with multiple conclusions by reformulating the tableau rules as block tableau rules and then converting these into the notation of Gentzen systems. The following table shows the block tableau rules on the left and the gentzen proof rules on the right.

T	F	L	R
γ $S, T(\forall x A)$ $S, T(A[t/x])$	δ $S, F(\forall x A)$ $S, F(A[a/x])$	∀L $H, \forall x A \vdash G$ $H, A[t/x] \vdash G$	∃L $H \vdash G, \forall x A$ $H \vdash G, A[a/x]$
δ $S, T(\exists x A)$ $S, T(A[a/x])$	γ $S, F(\exists x A)$ $S, F(A[t/x])$	∀R $H, \exists x A \vdash G$ $H, A[a/x] \vdash G$	∃R $H \vdash G, \exists x A$ $H \vdash G, A[t/x]$
The parameter t substituted for x can be chosen arbitrarily while a must be new			

Note that the same proviso applies to sequent proof rules as to tableau rules. Gentzen's original paper and most presentations of Gentzen systems use terms built from variables and function symbols instead of parameters. In that case a γ rule may substitute an arbitrary term t for x while the δ rule must choose a new (!) variable a.

16.5 First-Order Refinement Logic

The rules of first-order refinement logic can be extracted from those for multi-conclusioned sequents by dropping the extra conclusions, adopting a list notation, and adding a description of the evidence constructed by each rule. This leads to the following set of proof rules

Elimination (left)	Introduction (right)
`allL` i t $H, f{:}\forall x A, \Delta \vdash G$ ev = $g[f(t)/pf]$ $H, f{:}\forall x A, pf{:}A[t/x], \Delta \vdash G$ ev = $g[pf]$	$H \vdash \forall x A$ ev = `fun` $a \rightarrow pf[a]$ `allR` $H \vdash A[a/x]$ ev = $pf[a]$
`exL` i $H, z{:}\exists x A, \Delta \vdash G$ ev = `let` $z{=}(a, pf)$ `in` $g[a, pf]$ $H, pf{:}A[a/x], \Delta \vdash G$ ev = $g[a, pf]$	$H \vdash \exists x A$ ev = (t, pf) `exR` t $H \vdash A[t/x]$ ev = pf
t can be an arbitrary term while a must be a new variable	

Since refinement logic is based on a term language we have rephrased the proviso accordingly.[3] Note that the rule `allL` explicitly re-introduces the assumption $\forall x A$ in the subgoal. The reason for this is that univerally quantified formulas in the assumptions may have to be instantiated several in order to complete the proof. Any proof of `((∀x) Px)` $\Rightarrow$ `(Pa ∧ Pb)` must instantiate the variable `x` with both `a` and `b`. If `allL` would drop the assumption $\forall x A$, then some proof attempts would not succeed as they cannot show both `Pa` and `Pb`. Here is a proof of that statement in refinement logic.

[3] Terms are defined inductively: every variable x is a term and if f is an n-ary function symbol and $t_1, .., t_n$ are terms then $f(t_1, .., t_n)$ is a term. Constants are 0-ary function symbols applied to an empty list of terms and are written without the parentheses. Note that in most applications certain function symbols use a special (infix or other) syntax.

```
 ⊢ ((∀x) Px)  ⇒  (Pa ∧ Pb)            by impR
   (∀x) Px ⊢ Pa ∧ Pb                  by allL 1 a
    (∀x) Px, Pa ⊢ Pa ∧ Pb             by allL 1 b
     (∀x) Px, Pa, Pb ⊢ Pa ∧ Pb        by andR
 [1]  (∀x) Px, Pa, Pb ⊢ Pa            by axiom 1
 [2]  (∀x) Px, Pa, Pb ⊢ Pb            by axiom 2
```

Note that there is a differen RL proof for ((∀x) Px) ⇒ (Pa ∧ Pb) that uses `allL` only once in each branch of the proof and would succeed even if `allL` were to drop the univerally quantified formula. But the above example shows that `allL` would at least be irreversible. The fact that keeping the univerally quantified formulas is crucial for completeness is related to the use of a term structure instead of parameters. The formula `(((∀x) (P(x) ⇒P(f(x)))) ∧ P(a)) ⇒ P(f(f(a)))` can only be proven if we instantiate `x` with `a` and then with f(a). In this case both instances must occur in the same branch, since we need the first instance to prove P(f(a)) from P(a) and the second to get P(f(f(a))) from the P(f(a)) that we just proved.

The construction of evidence can be explained as follows:

`allR` In order to prove $\forall x A$ we have to prove $A[a/x]$ for an arbitrary (new) variable a. This will give us some proof evidence pf which will very likely depend on a. Since we must be able to do so without actually knowing a we will have constructed a function that given an arbittrary a will construct the evidence $pf[a]$. This function is sufficient evidence for the fact that A is true for all (arbitrary) x.

`allL` To prove a goal G by decomposing the assumption $\forall x A$ we introduce the assumption $A[t/x]$ for some term t that is given as parameter of the rule and prove G on that basis. This will give us some proof evidence g which may depend on the evidence pf for the truth of $A[t/x]$. Now if f is the function that describes the evidence for the assumption $\forall x A$, then $f(t)$ is evidence for $A[t/x]$ and can thus replace pf in the evidence $g[pf]$ for G.

`exR` In order to prove $\exists x A$ we have to prove $A[t/x]$ for a given term t. If pf is evidence for $A[t/x]$ then the combining this evidence with the information which term we used into a pair (t, pf) is evidence for the existence fact that A is true for some Element x.

`exL` To prove a goal G by decomposing the assumption $\exists x A$ we introduce the assumption $A[a/x]$ for some arbitrary (new) variable a (to indicate that we don't know for which specific Element a the assumption $A[a/x]$ holds) and prove G on that basis. This will give us some proof evidence g which may depend on the evidence pf for the truth of $A[t/x]$ and on the variable a. Now if z is a placeholder for the pair that describes the evidence for the assumption $\exists x A$, then z can be written as (a, pf) and a and pf can be used to construct $g[a, pf]$

Note that the above description links universal quantifiers to implication and existential quantifiers to conjunction and does not view them as generalized conjunction or generalized disjunction as it is commonly done. The reason is that we focus on the construction of evidence for truth instead of a possibly infinite (or even nondenumerable) combination of formulas.