9 Trees for Quantified Modal Logic

We now expand the method of semantic trees to cover quantified modal logic.

Trees can most directly be used to attempt to construct tK (or tT, tB, tS4, tS5, etc.) counterexamples, and thereby to test for tK (or tT, tB, tS4, tS25, etc.) validity, but can indirectly be used to test for other kinds of validity as well.

The process is more or less the same as what we had for propositional modal logic; we simply add rules dealing with the quantifiers, and identity.

9.1 Quantifier Rules

Note, these rules are applied *in the same world box.*

True ∀-statements: if you have both $\forall x\, Ax$ and Ec on the same branch in a given world box, write Ac on the branch in that world box as well.

$$\boxed{\begin{array}{c} \vdots \\ \forall x\, Ax \\ \vdots \\ \mathrm{E}c \\ \vdots \\ Ac \end{array}}$$

This rule needs to be applied as many times as you have constants on the tree branch, and may need to be reapplied if new constants are added (similar to the rule for true □-statements.)

False ∀-statements: if you have $\sim\forall x\, Ax$ on a tree branch in a given world box, write both Ec and $\sim Ac$ on the branch for that world box, where c is a *new* constant, not already on the tree.

$$\boxed{\begin{array}{c} \vdots \\ \sim\forall x\, Ax \\ \vdots \\ \mathrm{E}c \\ \sim Ac \end{array}}$$

Because of the definition of "∃", we get the following for existentially quantified statements (the difference between them is a lot like the corresponding differences for "◇" and "□"):

True ∃-statements: if you have $\exists x\, Ax$ on a tree branch in a given world box, write both Ec and Ac on the branch for that world box, where c is a *new* constant, not already on the tree.

$$\boxed{\begin{array}{c} \vdots \\ \exists x\, Ax \\ \vdots \\ \mathrm{E}c \\ Ac \end{array}}$$

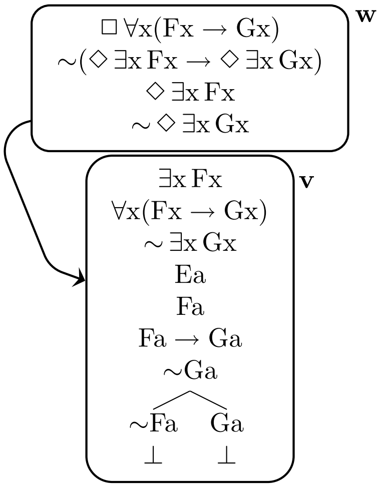
False ∃-statements: if you have both $\sim\exists x\, Ax$ and Ec on the same branch in a given world box, write $\sim Ac$ on the branch in that world box as well.

$$\boxed{\begin{array}{c} \vdots \\ \sim\exists x\, Ax \\ \vdots \\ \mathrm{E}c \\ \vdots \\ \sim Ac \end{array}}$$

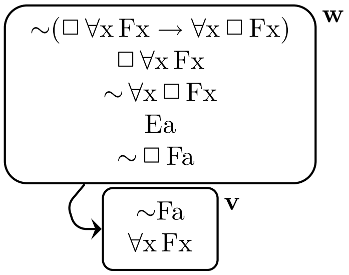
This may also be applied multiple times for as many constants as there are on the branch, and reapplied when new ones are added.

To constract trees for classical systems, simply eliminate the Ec step from either what you need, or what you get, as appropriate.

An example tree, showing $\Box\forall \mathrm{x}(\mathrm{Fx} \rightarrow \mathrm{Gx}) \vDash_{\mathrm{tK}} \Diamond\exists \mathrm{x}\,\mathrm{Fx} \rightarrow \Diamond\exists \mathrm{x}\,\mathrm{Gx}$:



The following instance of (CBF) is invalid: $\nvDash_{\mathrm{tK}} \Box\forall \mathrm{x}\,\mathrm{Fx} \rightarrow \forall \mathrm{x}\,\Box\,\mathrm{Fx}$.


$\sim(\Box\,\forall x\,Fx \to \forall x\,\Box\,Fx)$
$\Box\,\forall x\,Fx$
$\sim\forall x\,\Box\,Fx$
Ea
$\sim\Box\,Fa$
w
$\sim Fa$
$\forall x\,Fx$
v

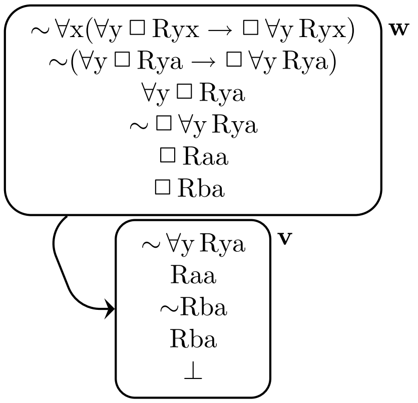
We do not get "Fa" in **v**, despite everything in **v** being F, since a may not exist in **v**!

Notice that if our domain were constant, a would have to exist in **v** since it exists at **w**. With free logic, however, this is not forced upon us.

If we were using the simplifications for classical quantification, we'd get "Fa" immediately and a closed tree.

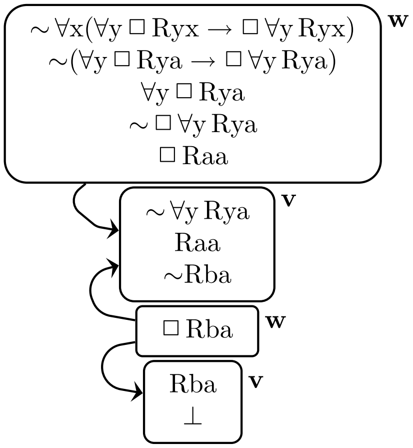
For reasons that are probably already obvious, it makes sense to do the rules that generate new constants (true ∃- and false ∀-statements) before applying or attempting to apply the rules for true ∀- and false ∃-statements.

Especially in trees for classical quantification systems, it is sometimes necessary to apply the rule for a true ∀- and false ∃-statements when a new constant has been introduced in *another* world box, possibly one below. The results are still placed in the same world box as the true ∀- or false ∃-statement. Consider this tree for $\vDash_{\mathrm{qK}} \forall x(\forall y\,\Box\,Ryx \to \Box\,\forall y\,Ryx)$


$\sim\forall x(\forall y\,\Box\,Ryx \to \Box\,\forall y\,Ryx)$
$\sim(\forall y\,\Box\,Rya \to \Box\,\forall y\,Rya)$
$\forall y\,\Box\,Rya$
$\sim\Box\,\forall y\,Rya$
$\Box\,Raa$
$\Box\,Rba$
w
$\sim\forall y\,Rya$
Raa
$\sim Rba$
Rba
$\perp$
v

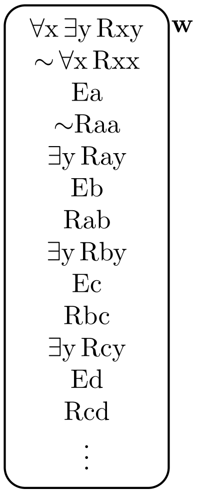
In the result, it may look as if we did the rule for the true ∃-statement to a letter that wasn't new. This is not so. The requirement that it be new means it must have been new *when* we introduced it. It's a temporal, not a spatial restriction. Here, the "□ Rba" in world **w** was introduced only *after* we put "b" into world **v**. We *then* put "Rba" into **v**, which closed the tree.

It may be less confusing to use the "continuation" method instead, although that may lead to unusual looking accessibility arrows. E.g., with the earlier example:


$\sim\forall x(\forall y\,\Box\,Ryx \to \Box\,\forall y\,Ryx)$
$\sim(\forall y\,\Box\,Rya \to \Box\,\forall y\,Rya)$
$\forall y\,\Box\,Rya$
$\sim\Box\,\forall y\,Rya$
$\Box\,Raa$
w
$\sim\forall y\,Rya$
Raa
$\sim Rba$
v
$\Box\,Rba$
w
Rba
$\perp$
v

(This is the first time we've seen a one-sided arrow go *up*.)

We saw an example or two of K4-trees or S5-trees in propositional modal logic that could not be completed, because they would go on forever. There, the problem was that new worlds would continue to be spawned. In quantified logic, a tree can end up going on forever if new constants are foreover introduced:

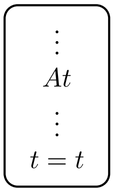


This means that trees are not a decision procedure for quantified logic (modal or not). Indeed, it is *impossible* to invent an algorithm for testing the validity of any argument in predicate logic that is guaranteed always to terminate in a finite number of steps. (This is actually a corollary of Gödel's famous incompleteness theorems known as *Church's theorem*—not to be confused with Church's thesis, or the Church-Turing thesis, which is different, although related. I cover the proof of Church's theorem in my Math Logic I course.) It also marks a point of departure from propositional modal logic, where there are decision procedures (even if trees weren't it).

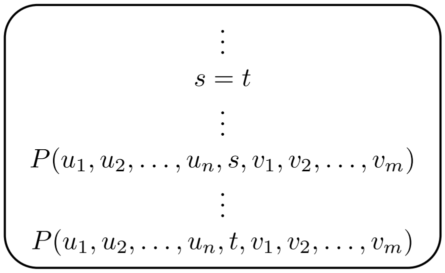
9.2 Identity Tree Rules

The rules for dealing with identity statements match the deductive system inference rules very closely.

- Insert $t = t$ into every world box in which the term t appears. (Note this may mean inserting it into multiple world boxes.)



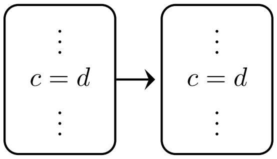
- If $s = t$ appears in a given world box, and so does an atomic statement, $P(u_1, u_2, \ldots, u_n, s, v_1, v_2, \ldots, v_m)$, then write in $P(u_1, u_2, \ldots, u_n, t, v_1, v_2, \ldots, v_m)$ unless it already occurs in the world box in question.



⋮
$s = t$
⋮
$P(u_1, u_2, \ldots, u_n, s, v_1, v_2, \ldots, v_m)$
⋮
$P(u_1, u_2, \ldots, u_n, t, v_1, v_2, \ldots, v_m)$

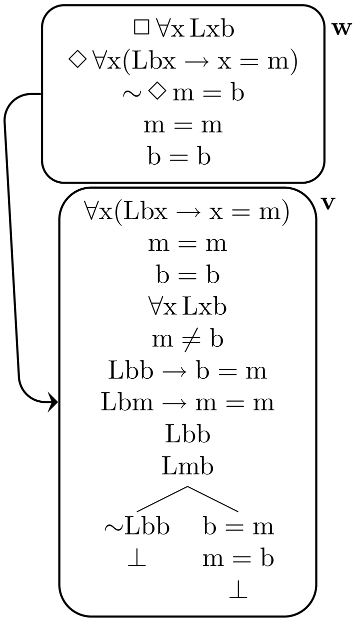
(Note that strictly speaking, substitutions on identicals go from left to right. However, the symmetry of the identity statement will appear thanks to the first identity rule, which will force substitutions the other way as well. We shall allow ourselves to overlook this complication.)

- If we wish to construct a tree assuming (RC), or a rigid interpretation of the constants, whenever $c = d$, both constants, occurs in one world box on a branch, we insert it into all other world boxes on the branch in question (using continuations if there is branching in between).



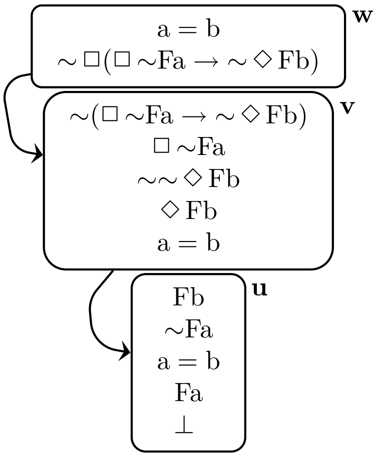
(either direction)

An example tree for:
$\Box \forall x\, Lxb, \Diamond \forall x(Lbx \to x = m) \vDash_{qK} \Diamond m = b$



Although I did all of them above, generally, it is not necessary to do each $t = t$ step unless it leads straightaway to a contradiction, or is needed for symmetry. Indeed, one may think of the rule instead as to insert $\perp$ if you get something of the form $t \neq t$

Here's a tree for rigid constants.



Since "a" and "b" are both constants, we treated them as names of the same entity in all worlds, because our premises gave us that they are the same in one world.

We would also be able to apply this rule in the other direction (even against the "grain" of an accessibility relation arrow): recall that (RC) also assumes that if $c = d$ is not true in the actual world, then it is *necessarily* untrue.