

PY1003 Introduction to Logic  
Lecture12

Remember that when we apply the tree rule for  $\forall$  to a universally quantified sentence that sentence is not used up. We can use it again.

$$\begin{array}{l} \forall xFx \quad \sqrt{a} \quad \sqrt{b} \\ | \\ Fa \\ | \\ Fb \end{array}$$

You can keep going forever with this process, provided the branch does not close at any point.

A Problem?

In order to prove invalidity or consistency using a tree, we have to show that we have applied *every rule we can* and that there are still open branches. But if we start with a universally quantified sentence, we've *never* applied every rule we can, since we can always go back and instantiate the sentence again using a new individual letter. So how can we prove invalidity or consistency using a tree when we are dealing with universally quantified sentences?

The answer is that, for our purposes, we can regard a universally quantified sentence that appears on a branch as "finished" once we have instantiated it using every individual letter that appears on the branch (including, if necessary, a new individual letter introduced especially for the purpose of instantiating that sentence). The reason this is acceptable is that if the branch is going to close at all, it will close under these circumstances. So if we apply these rules and it doesn't close, then we've shown that it will never close.

Sometimes we have a choice of instantiations for a universally quantified sentence. That is, we have more than one 'old' individual letter which we could use to instantiate the sentence. Consider the tree which begins with these three sentences:

$$\begin{array}{l} \forall xFx \\ Gb \\ \neg Fa \end{array}$$

We want to apply the rule for  $\forall$  to the first line, but we have a choice of two old names to use: a and b. Which should we choose? Either would be acceptable. But look what happens if we choose b:

$$\begin{array}{l} \forall xFx \quad \sqrt{b} \\ \neg Fa \\ Gb \\ | \\ Fb \end{array}$$

And look what happens if we choose a:

$$\begin{array}{l} \forall xFx \quad \quad \quad \sqrt{^a} \\ \neg Fa \\ Gb \\ | \\ \underline{Fa} \end{array}$$

Since instantiating with a closes the branch, this is clearly the letter to choose if we want to get our tree finished as quickly as possible. If we instantiate with b first, this is not a mistake; we can come back and instantiate with a later, because the first line is not used up:

$$\begin{array}{l} \forall xFx \quad \quad \quad \sqrt{^b} \quad \sqrt{^a} \\ \neg Fa \\ Gb \\ | \\ Fb \\ | \\ \underline{Fa} \end{array}$$

We just save ourselves a bit of time by spotting that we should instantiate with a in the first place.

Sometimes you *have* to instantiate the same sentence twice in order to get all the branches to close.

Example:

Is the set  $\{\forall xFx, \neg Fa \vee \neg Fb\}$  consistent?

$$\begin{array}{l} \forall xFx \quad \quad \quad \sqrt{^a} \quad \sqrt{^b} \\ \neg Fa \vee \neg Fb \quad \quad \quad \sqrt{ } \\ | \\ Fa \\ | \\ Fb \\ / \quad \backslash \\ \underline{\neg Fa} \quad \underline{\neg Fb} \end{array}$$

All the branches close, so the set is inconsistent. But note that we needed to use both instantiations in order to close all the branches.

There are just two more new tree rules to learn for Predicate logic. They depend on the *interdefinability* of the quantifiers. Remember:

1.  $\forall xFx$  iff  $\neg \exists x \neg Fx$
2.  $\exists xFx$  iff  $\neg \forall x \neg Fx$

From 2 we know that:

3. If  $\neg \forall x \neg Fx$ , then  $\exists x Fx$ .

This means that:

4. If  $\neg\forall x\neg Fx$ , then  $\exists x Fx$ .

Because  $\neg\neg Fx$  is the same as  $Fx$ , this means that:

5. If  $\neg\forall x Fx$ , then  $\exists x\neg Fx$ .

Another way of seeing why 5 is true is to note that if it is not the case that everything is F, then it must be that something is not F.

THE RULE FOR  $\neg\forall$ :

$$\frac{\neg\forall x Fx}{\exists x\neg Fx}$$

NB: we tick off sentences in the ordinary way when we have applied this rule to them. The only sentences which get a special kind of tick are non-negated universally quantified sentences.

From 1 above we know that:

6. If  $\neg\exists x\neg Fx$ , then  $\forall x Fx$ .

This means that:

7. If  $\neg\exists x\neg\neg Fx$ , then  $\forall x\neg Fx$ .

Because  $\neg\neg Fx$  is the same as  $Fx$ , this means that:

8. If  $\neg\exists x Fx$ , then  $\forall x\neg Fx$ .

Another way of seeing why 8 is true is to note that if it is not the case that something is F, then it must be that everything is not F.

THE RULE FOR  $\neg\exists$ :

$$\frac{\neg\exists x Fx}{\forall x\neg Fx}$$

We can now construct a tree to test the following argument for validity:

David is rich

Therefore, someone is rich

Translating, we get  $Ra \vdash \exists x Rx$ . And we construct the tree as follows:

$$\begin{array}{l} Ra \\ \neg\exists x Rx \quad \checkmark \\ | \\ \forall x\neg Rx \quad \checkmark^a \\ | \\ \underline{\neg Ra} \end{array}$$

Since there are no open branches, the argument is shown to be valid.

## Using the quantifier rules together

Consider the following argument:

Someone is happy. Everyone who is happy is rich. Therefore someone is rich.

Clearly it is valid. Translating into Predicate, we get:

$$\exists xHx, \forall x(Hx \rightarrow Rx) \vdash \exists xRx$$

We begin our tree:

$$\begin{array}{l} \exists xHx \\ \forall x(Hx \rightarrow Rx) \\ \neg \exists xRx \end{array}$$

Now what?

We could instantiate the existential on the first line, or we could instantiate the universal on the second line, or we could apply the rule for  $\neg \exists$  to the third line.

A good rule of thumb is to apply the rules for negated quantified formulae first. They are non-branching rules, and getting them out of the way helps to clarify what you're dealing with.

So:

$$\begin{array}{l} \exists xHx \\ \forall x(Hx \rightarrow Rx) \\ \neg \exists xRx \quad \checkmark \\ | \\ \forall x \neg Rx \end{array}$$

Now we have one existentially quantified sentence and two universally quantified sentences waiting for rules to be applied to them.

Another rule to remember is that we should instantiate existentially quantified sentences before instantiating universally quantified sentences. The reason for this is that instantiating existentials always introduces new individual letters onto the branch, and we may need to use these letters to instantiate the universals later on.

Remember that we should, wherever possible, only use *old* letters to instantiate universals. In our example, we have no old letters around to use to instantiate our universal quantifiers. We *could* just introduce a new letter specifically for the purpose of instantiating one of the universals, but we should *only do that if we have to*. And we don't have to here: instantiating the existential first will introduce a new letter, and we'll then be able to use that letter later on to instantiate the universals.

So:

$$\begin{array}{l}
 \exists x Hx \quad \checkmark \\
 \forall x (Hx \rightarrow Rx) \\
 \neg \exists x Rx \quad \checkmark \\
 | \\
 \forall x \neg Rx \\
 | \\
 Hc
 \end{array}$$

Next we instantiate the two universals using the (now) old letter c, and the all the branches will close:

$$\begin{array}{l}
 \exists x Hx \quad \checkmark \\
 \forall x (Hx \rightarrow Rx) \quad \checkmark^c \\
 \neg \exists x Rx \quad \checkmark \\
 | \\
 \forall x \neg Rx \quad \checkmark^c \\
 | \\
 Hc \\
 | \\
 Hc \rightarrow Rc \quad \checkmark \\
 | \\
 \neg Rc \\
 / \quad \backslash \\
 \underline{\neg Hc} \quad \underline{Rc}
 \end{array}$$

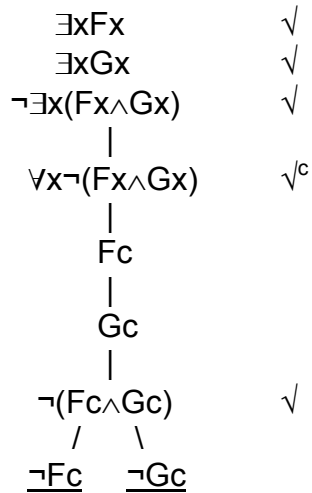
Note that if you have two existentially quantified sentences in your tree, you should introduce a new letter for each of them.

Example:

Is this argument valid?  $\exists x Fx, \exists x Gx \vdash \exists x (Fx \wedge Gx)$

$$\begin{array}{l}
 \exists x Fx \quad \checkmark \\
 \exists x Gx \quad \checkmark \\
 \neg \exists x (Fx \wedge Gx) \quad \checkmark \\
 | \\
 \forall x \neg (Fx \wedge Gx) \quad \checkmark^c \quad \checkmark^d \\
 | \\
 Fc \\
 | \\
 Gd \quad \text{c is already an 'old' letter now} \\
 | \\
 \neg (Fc \wedge Gc) \quad \checkmark \\
 \neg (Fd \wedge Gd) \quad \checkmark \\
 / \quad \backslash \\
 \underline{\neg Fc} \quad \neg Gc \\
 \quad \quad / \quad \backslash \\
 \quad \quad \neg Fd \quad \underline{\neg Gd}
 \end{array}$$

The tree has an open branch and we have applied all the rules we need to (including instantiating the universal using every individual letter that's appears in the tree). So the argument is shown to be invalid. But look what would have happened if we had allowed ourselves to instantiate both the existentially quantified sentences using the same letter:



Things to remember:

1. Apply rules for negated quantified sentences early on.
2. Don't use new individual letters to instantiate universals unless you have to.
3. Instantiate existentials before universals.
4. Always use a new individual letter for *each* existential instantiation.