MAU22C00 Lecture 9

John Stalker

Trinity College Dublin

Informal arguments in first order logic

A statement in first order logic is said to be *valid* if it is true in any (set-based) interpretation. Valid statements in first order logic play the same role as tautologies in zeroeth order logic.

There is no equivalent to truth tables. We don't know what values the variables could take and there are probably infinitely many of them.

We can still prove statements are valid by case by case analysis though, and can still use tableaux as a bookkeeping device.

An informal argument

$[([\forall x.(f x)] \wedge \{\exists x.[(f x) \supset (g x)]\}) \supset (\exists x.\{g x\})]$ is valid. We can see this as follows.

For it to be false in some interpretation $[([\forall x.(f x)] \wedge \{\exists x.[(f x) \supset (g x)]\})]$ would need to be true in that interpretation and $(\exists x.\{g x\})$ would need to be false.

Then $[\forall x.(f x)]$ and $\{\exists x.[(f x) \supset (g x)]\}$ would both be true.

$\{\exists x.[(f x) \supset (g x)]\}$ asserts the existence of at least one x such that $[(f x) \supset (g x)]$. Let a be such a value of x. Then $[(f a) \supset (g a)]$.

We have $[\forall x.(f x)]$, i.e. that $(f x)$ for every value of x. That includes a so $(f a)$.

$(\exists x.\{g x\})$ is false, so $(g a)$ is also false.

For $[(f a) \supset (g a)]$ to be true we need $(f a)$ to be false or $(g a)$ to be true, but we've already seen that $(f a)$ is true and $(g a)$ is false.

The assumption that $[([\forall x.(f x)] \wedge \{\exists x.[(f x) \supset (g x)]\}) \supset (\exists x.\{g x\})]$ is false in some interpretation leads to a contradiction, so it's true in every interpretation, i.e. the statement is valid.

A tableau

The same argument for the validity of $[([\forall x.(fx)] \land \{\exists x.[(fx) \supset (gx)]\}) \supset (\exists x.\{gx\})]$ in tableau form is

$$
\begin{array}{r|l}
 & [([\forall x.(fx)] \land \{\exists x.[(fx) \supset (gx)]\}) \supset (\exists x.\{gx\})] \\
([\forall x.(fx)] \land \{\exists x.[(fx) \supset (gx)]\}) & \\
 & (\exists x.\{gx\}) \\
[\forall x.(fx)] & \\
\{\exists x.[(fx) \supset (gx)]\} & \\
[(fa) \supset (ga)] & \\
(fa) & \\
 & (ga)
\end{array}
$$

$$
\begin{array}{l|c|c|r}
 & \underline{(fa)} \quad\quad\quad\quad & \quad\quad\quad\quad \underline{(ga)} &
\end{array}
$$

Figure 1: A tableau for $[([\forall x.(fx)] \land \exists x.[(fx) \supset (gx)]) \supset (\exists x.gx)]$

Tableaux rules for first order logic

There are four new rules, depending on which quantifier is involved and which side of the bar it's on, three of which we used above.

Every rule involves removing the quantifier and following variable, substituting a parameter for all free occurences of that variable, and putting the resulting statement on the same side of the line as the original one.

In two cases, where we have an $\forall$ on the left or an $\exists$ on the right, we can replace the variable by *any* parameter.

In the other two cases, where we have an $\forall$ on the right or an $\exists$ on the left, we can replace it with any parameter *which has not been used previously on that branch*.

This makes sense if you think about it. It's easier to see for statements on the left. The arguments for statements on the right are similar, but with $\forall$ and $\exists$ swapped.

A statement which holds for all values holds for any value, including ones we've already given names to or ones we haven't named yet.

If it holds for some value then we can give a name to such a value but we can't assume that value is the same as one we've named earlier, so we need a new name.

Our example, again

We used three of these rules above.

"$\{\exists x.[(fx) \supset (gx)]\}$ asserts the existence of at least one x such that $[(fx) \supset (gx)]$. Let a be such a value of x. Then $[(fa) \supset (ga)]$."

This is the rule for an $\exists$ on the left, one of the restricted ones. We have to introduce a new parameter, not reuse a previous one. In this case there weren't any previous ones so we would have had to anyway.

In the tableau version we had a $\{\exists x.[(fx) \supset (gx)]\}$ on the left and wrote $[(fa) \supset (ga)]$ below it on the left.

"We have $[\forall x.(fx)]$, i.e. that (fx) for every value of x. That includes a so (fa)."

This is the rule for an $\forall$ on the left. This is an unrestricted one, so we can reuse the parameter a. We don't have to, but if we didn't then we wouldn't get the contradiction we're looking for.

In the tableau we put (fa) on the left, below $[\forall x.(fx)]$.

Our example, continued

"$(\exists x.\{gx\})$ is false, so (ga) is also false."

This is the rule for an $\exists$ on the right, which is also an unrestricted case. Again, we can reuse the parameter a. We don't have to, but if we didn't then once again we wouldn't get the contradiction we're looking for.

On the tableau we put (ga) on the right, below $(\exists x.\{gx\})$.

Note that none of the quantifier rules ever involve branching. That only happens for the binary Boolean operators, and only when they're on one side of the line.

Does this work?

In *zeroeth order logic* we have the following useful properties:

- The procedure will eventually end no matter what choices you make[1]
- If the statement is a tautology all the branches will close, no matter what choices you make,[2]
- If the statement is not a tautology then there will be at least one branch where you exhaust all the options[3] without being able to close it. This will give you a counterexample.

In first order logic you might hope for the same properties, just replacing tautologies by valid statements.

Unfortunately none of them are true.

In our example we could have just kept using our quantified statements, replacing the variable by a new parameter each time.

---

[1] assuming you're not deliberately obtuse.

[2] assuming again you're not deliberately obtuse.

[3] other than rewriting the same statements over and over again

What does work

The most you can say in first order logic is that if a statement is valid then *some* set of choices (of the order in which to process statements, the parameters to substitute for variables, etc.) will lead to all branches closing.

There are two options to avoid getting stuck:

- make our tableau algorithm fully deterministic, but very carefully
- make our tableau algorithm fully non-deterministic, i.e. every time we have more than one option explore the consequences of each possible choice

The first is better if you want to prove the validity of an actual statement.

The second is better if you want to prove the completeness of first order logic.

The second option has an interpretation in terms of non-deterministic computations, which I'll talk about next time.