

24/03/2006, Dr Marcus Rossberg, mr30@st-and.ac.uk
 office hours: Thursdays, 10-12, (*Arché* building, College St, 19.4)

PY1003

Lecture 13

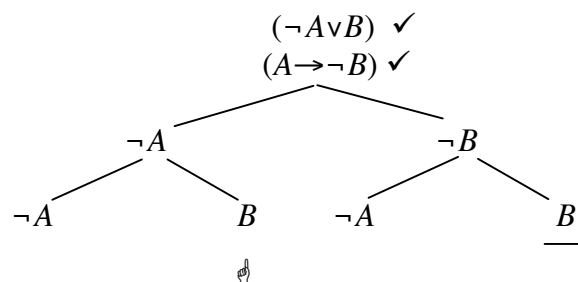
Models and Counterexamples

Recap: *What is a model in propositional logic?*

A *model* of a set of sentences is an assignment of truth-values to sentence letters that makes all sentences in the set true. Providing a model for a set of sentences shows that this set is *consistent* – all sentences can be simultaneously true.

Models can be read off a tree by following an open branch from a leaf to the root. Assign ‘T’ to all atomic sentences that occur in the branch, and ‘F’ to all the negated atomic sentences that occur in it.

Example: Consider the set $\{(\neg A \vee B), (A \rightarrow \neg B)\}$:



The open branches show that the set is consistent. The model for the set read off the branch pointed out by the hand-icon (‘ \uparrow ’) is:

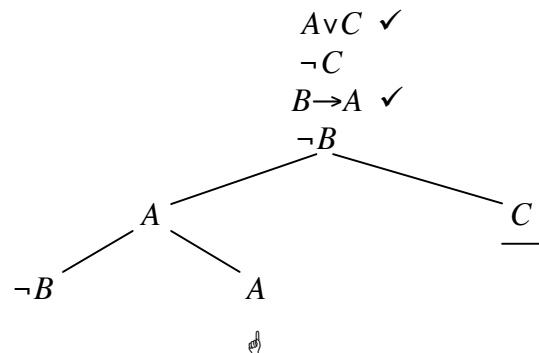
$$\begin{aligned} I(A) &= F \\ I(B) &= T \end{aligned}$$

What is a counterexample or countermodel in propositional logic?

Recall that a sentence, called the conclusion, *follows* from some sentences, called the premises, iff the set that contains the premises and the negation of the conclusion is *inconsistent*, i.e. when it has *no model*.

If, on the other hand, the set is consistent, i.e. has a model, the purported conclusion does not follow from the premises. The model of the set is called a *countermodel* to the inference, or a *counterexample* to the argument.

Example: Test whether $A \vee C, \neg C, B \rightarrow A \vdash B$



The tree has open branches; the inference is not valid.

The countermodel that can be read off the branch pointed out is:

$$\begin{array}{l}
 I(A)=T \\
 I(B)=F \\
 I(C)=F
 \end{array}$$

Models in Predicate Logic

The notion of a model extends in a straightforward way from propositional to predicate logic.

The atomic sentences of predicate logic look different, however. They consist of predicates and names. If we want to specify a model for a set of sentences of predicate logic it does *not* suffice to just say whether they are interpreted as true or false.

In propositional logic, the truth-value of the complex sentences is a function of the atomic sentences that constitute it. This is not the case in predicate logic, however: the quantifiers forbid this treatment, as we will see in a minute.

Nevertheless, there is a way to specify conditions under which all sentences of a consistent set are true, and these can as easily be read off a tree as in propositional logic.

Consider this set with only one sentence as a member as an example: $\{\forall x Hx\}$

The short tree to test for consistency looks like this:

$$\begin{array}{c}
 \forall x Hx \quad \checkmark a \\
 Ha
 \end{array}$$

Here we can stop – no further application of the rule for ‘ \forall ’ will make the branch close since except for ‘ a ’ no other names are in the only branch.

Just to specify that ‘ Ha ’ has to be true will not do as a model for $\{\forall x Hx\}$. For even if ‘ Ha ’ is true, ‘ $\forall x Hx$ ’ could still be false, e.g. if ‘ Hb ’ was false! Nothing in $\{\forall x Hx\}$,

however, requires us to assume that there exists something other than a . We *always* assume that at least *one* thing exists, though.

Say, ' H ' stood for 'is happy' and ' a ' for Anna; to show that $\{\forall xHx\}$ is consistent by providing a model we only have to say that only Anna exists (according to this interpretation or model), and that she is happy.

This is enough to guarantee that ' $\forall xHx$ ' is true. One interpretation that makes every sentence in the set true is sufficient to show that it is consistent (this is how consistency is defined.)

We want a *formal* way, though, to construct models for predicate logic, one that can simply be read off a tree and ties in with propositional logic. Here is the short tree again:

$$\begin{array}{l} \forall xHx \quad \checkmark a \\ Ha \end{array}$$

Since it does not matter what ' a ' stands for, we just leave it at that, and merely specify that a is the only thing that exists. We say that we specify the *domain* of the model to contain a and only a , since ' a ' is the only name that occurs on the branch. We represent the domain by a set that contains exactly those things we assume to exist. In our case at hand, we write:

$$D=\{a\}$$

Now we provide an *interpretation* for *all* the predicates in the *entire* tree, not just the branch we are considering (in this case only ' H ', anyway).

We *could* then go about and specify for all the sentences that occur on a branch, whether they are to be true or false – there is a problem with this, however.

Imagine providing a model for a set of sentences that assumes that *infinitely many* things exist. We would have to write down for *infinitely many* sentences whether they are true or false to give the model. (Nobody has that much time... in particular not in an exam!)

To prevent this difficulty from occurring, we introduce a handy notion:

The extension of a predicate

We say that the extension of a predicate contains all and only those things that the predicate is true of.

In our case, we can determine what things a predicate has to be true of in a model, by just reading off the tree what the true atomic sentences are. In our little tree:

$$\begin{array}{l} \forall xHx \quad \checkmark a \\ Ha \end{array}$$

this is just ' Ha '. Thus, the extension of H contains a , and only a , because ' H ' is true of a and nothing else.

In the model we are providing, we can represent the extension of a predicate again as a set containing the things the predicate is true of. Thus, we specify:

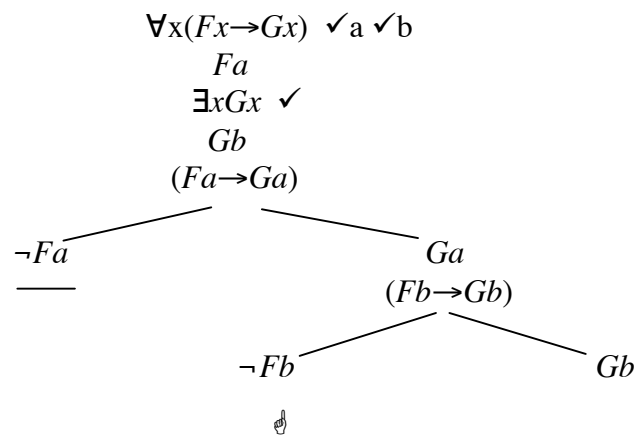
$$I(H) = \{a\}$$

A model of predicate logic consists of a domain and an interpretation.

To construct a **model** for $\{\forall x Hx\}$ we specified the **domain** as $D = \{a\}$, and the **interpretation** as $I(H) = \{a\}$.

Example: Test the following set for consistency:

$$\{\forall x(Fx \rightarrow Gx), Fa, \exists x Gx\}$$



There are open branches in the tree, so the set is consistent. The model that exhibits this, read off the branch pointed out, is:

$$D = \{a, b\}$$

$$I(F) = \{a\}$$

$$I(G) = \{a, b\}$$

On this branch ' Fb ' occurs with a negation symbol in front of it. We hence do not include b in the extension of F : ' F ' is *false* of b .

Had we read the model off the rightmost branch of the tree above, ' Fb ' would not have occurred at all.

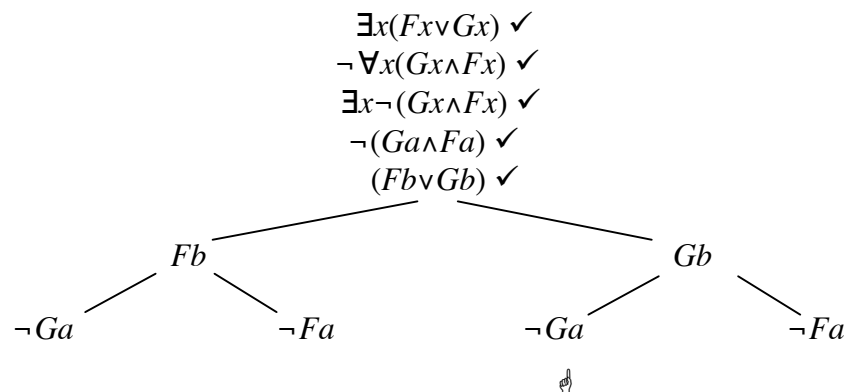
Let us adopt as a *rule of thumb* not to include the respective objects in the extension of the relevant predicate in such a case – to save ink – although it would make no difference: you get a model for the set either way.

Countermodels in Predicate Logic

Just like in propositional logic, a countermodel for an inference, or a counterexample to an argument, is a model of the set that contains the premises and the negation of the purported conclusion.

Example: Is the following inference valid? If not, provide a counterexample.

$$\exists x(Fx \vee Gx) \vdash \forall x(Gx \wedge Fx)$$



There are open branches in the tree, so the inference is not valid. The countermodel that exhibits this, read off the branch pointed out, is:

$$D = \{a, b\}$$

$$I(G) = \{b\}$$

$$I(F) = \{ \} = \emptyset$$

(‘ \emptyset ’ is the common symbol that is used to designate the empty set.)

Exercises

Are the following inferences valid? If not, provide a counterexample.

(1) $\exists x(Fx \vee Hx), Ha \vdash \forall x(Fx \rightarrow Hx)$

(2) $\forall x(Fx \rightarrow Gx), \forall x(Gx \rightarrow Hx), Fa \vdash Hc$

(3) $\exists xFx, Fa, Fb \vdash \forall xFx$

(4) $\forall x(Fx \rightarrow Gx), Fa, \neg Ga \vdash \exists x(\neg Fx \vee ((Gb \wedge Fa) \rightarrow Hx)) \wedge \forall y(Fy \rightarrow Gy)$