

11/04/2006, Dr Marcus Rossberg, mr30@st-and.ac.uk
office hours: Thursdays, 10-12, (*Arché* building, College St, 19.4)

PY1003

Lecture 14

Many-Place Predicates

Recap: What is a model in predicate logic?

A *model* of a set of sentences in predicate logic consists of a *domain* and an *interpretation*. The domain is represented by a set that contains all and only the things that have to be assumed exist in the model. In the interpretation the *extensions* of the predicates figuring in the sentences are specified. The extension of a predicate contains all and only the things that the predicate is true of. In the model the extension is represented by a set containing these things.

We can read off what has to go into the *domain* from the usual tree with which we test for consistency. All and only the things whose names occur in the open branch we read the model off are in the domain.

Similarly, we read the *interpretation* off the tree. The extensions of the predicates (according to the model) contain all and only the things whose names occur in the (not negated) open sentences of that branch.

To give a complete interpretation, we also have to specify the extensions of the predicates that do not occur in the branch but do occur somewhere in the set of sentences we test for consistency – should there be such predicates. Since it does not matter for the model what the extension of such predicates is, we adopted as a *convention* to say that these predicates are assigned an empty extension, represented by the *empty set*, \emptyset .

Specifying a model like this suffices to guarantee that all the sentences in the set are true under that interpretation.

What is a counterexample in predicate logic?

A *counterexample* to an argument or a *countermodel* to a purported inference in predicate logic is a model of the set that contains the premises and the negation of the purported conclusion of the argument.

It is read off the tree in exactly the same way as described before.

Relations in Predicate Logic

The grandfather of predicate logic, Aristotle, designed a logic, called *syllogistic*, that was in some respects similar to the predicate logic we have got so far. One similarity is the lack of being able to deal with arguments like the following one:

All horses are animals.

All horse-heads are animal-heads.

The solution, how to formalise the above, and show that the inference is valid, will have to wait until next week, however.

Note, though, that taking ‘*H*’ for ‘is a horse’, ‘*A*’ for ‘is an animal’, ‘*F*’ for ‘is a horse-head’ and ‘*G*’ for ‘is an animal-head’ won’t do. The inference would read:

$$\forall x(Hx \rightarrow Ax) \vdash \forall x(Fx \rightarrow Gx)$$

This is certainly not valid.

What the wanted formalisation involves are *many-place predicates*, which stand for *relations* in much the same way *one-place predicates* – those predicates we have encountered so far – stand for *properties*.

Syntactically, an expression containing a many-place predicate might look like this:

$$Rab$$

We can think of it as saying that *a* relates to *b* in a certain way, specified by the two-place (or *binary*) predicate *R*.

If we, for example, want to formalise ‘Bill loves Adam’ into predicate logic using the two-place predicate ‘*L*’ to stand for ‘loves’, we get:

$$Lba$$

Formalising ‘Adam does not love Bill’ we get:

$$\neg Lab$$

The *order* of the names, in this case ‘*a*’ and ‘*b*’, is of **vital importance**. This, however, is no difference from ordinary language: ‘Gill hates Jules’ says something very different from ‘Jules hates Gill’.

When we formalise ordinary language sentences into formal predicate logic, we want to use as few different predicate letters as possible while capturing the intended meaning of the sentence. This way we try to carve out as much logical structure as possible. We will not use different two-place predicates for ‘loves’ and ‘is in love with’, for example, otherwise, the blatantly valid inference from ‘Sue loves Mary’ to

‘Sue is in love with Mary’ would fail (assuming these are synonymous). We might formalise it as:

$$Lsm \vdash Ism$$

This does not look like a valid inference.

The same applies for ordinary language sentences which are in *passive voice*. ‘Sue loves Mary’ says the same as ‘Mary is loved by Sue’. Should we pick two different two-place predicates for ‘loves’ and ‘is loved by’ we would run into the same trouble as above. We will formalise ‘Mary is loved by Sue’ in the same way we formalise ‘Sue loves Mary’, using ‘*L*’ for ‘loves’, viz.:

$$Lsm$$

Here, again, you can see how important it is to get the order of the names right.

In ordinary language we can talk about relations that have more than two places, of course, and it should be obvious how we accommodate that in the language of predicate logic, too.

‘St Andrews lies between Edinburgh and Aberdeen’, for instance, can be formalised using ‘*B*’ for ‘... lies between ... and ...’ (a three-place predicate), ‘*s*’ for ‘St Andrews’, ‘*e*’ for Edinburgh, and ‘*a*’ for ‘Aberdeen’, as:

$$Bsea$$

We often talk of *argument places* of predicates. An ordinary one-place predicate has one argument place that is filled by a name (or variable), like the ‘*a*’ in ‘*Fa*’. Two-place predicates have two argument places, and so on. In this way we can talk *about* many-place predicates: we can say that in the sentence above ‘*e*’ occurs in the second argument place of the (three-place) relation ‘*B*’, or that the third argument place is filled by ‘*a*’.

Note: ‘Argument’ here is not to be confused with an inference called an argument!

Ordinary language examples for relations involving predicates with more than three argument places are harder to find.

Here is one with four argument places:

Moscow is as far east of Glasgow as New York is of L.A.

If we use the four-place predicate ‘*E*’ to stand for ‘... is as far east of ... as ... is of ...’, and ‘*m*’ as a name for Moscow, ‘*g*’ for Glasgow, ‘*n*’ for New York, and ‘*l*’ for L.A., we can formalise the sentence above as:

$$Emgnl$$

More than four argument places do not seem to occur in English predicates, unless we resort to mathematical examples. ‘The line that goes through the points a and b and the line that goes through the points c and d intersect in point e ’, for instance, is a five-place predicate – with the five places being occupied by the names for the five points.

There is no limit to the places a predicate can have, but the number of places is always fixed. Once we decided what a many-place predicate stands for, we have to stick to that. This is also the case for the number for places it has.

A predicate letter cannot stand for a three-place relation and also for a two-place relation in the same argument.

Often, the number of places a predicate has, is indicated by a superscript. The last example above, for instance, would then read:

$$R^4mgnl$$

For easier legibility we dispense with this convention here. We can figure out how many places the predicate has by counting the names that follow it.

You will sometimes also see that for two-place predicates the predicate letter is written between the names. So, instead of:

$$Rab$$

sometimes

$$aRb$$

is written. It means *exactly the same*, however. It is a mere *notational variant*. We do not follow this usage here, as it does not extend in any natural way to predicates with more than two argument places.

Quantification

Like with one-place predicates, we can also use quantifiers with many-place predicates. If we replace names by variables, we can bind these with quantifiers.

‘Every student loves Marcus’, for example, can be formalised using ‘ S ’ for ‘is a student’, ‘ L ’ for ‘loves’, and ‘ m ’ for ‘Marcus’ as:

$$\forall x(Sx \rightarrow Lxm)$$

Again, we can see that it is very important to keep the order of the places in the many-places predicate right. The sentence above says something very different from:

$$\forall x(Sx \rightarrow Lmx)$$

This would be the formalisation of ‘Marcus loves every student’. Equally, ‘Marcus is hated by some professors’ (with ‘*H*’ for ‘hates’, ‘*m*’ for ‘Marcus’, and ‘*P*’ for ‘is a professor’):

$$\exists x(Px \wedge Hxm)$$

Is certainly different from ‘Marcus hates some professors’:

$$\exists x(Px \wedge Hmx)$$

Trees

We *do not need any new rules* for the trees if we want to use many-place predicates. ☺
The rules we have apply to sentences containing one-place predicates in just the same way as they apply to sentences containing many-place predicates.

Example: Is the following argument valid? $Lab, \forall xFx \vdash \exists x(Lxb \wedge Fx)$

$$\begin{array}{c}
 Lab \\
 \forall xFx \quad \checkmark a \quad \checkmark b \\
 \neg \exists x(Lxb \wedge Fx) \quad \checkmark \\
 \forall x \neg(Lxb \wedge Fx) \quad \checkmark a \\
 Fa \\
 Fb \\
 \neg(Lab \wedge Fa) \quad \checkmark \\
 \hline \neg Lab \qquad \qquad \qquad \neg Fa \\
 \hline
 \end{array}$$

All the branches are closed. The argument is valid.

Models and Counterexamples

Concerning the models of sets of sentences and counterexamples to an invalid argument, nothing *much* changes either.

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

$$\neg \forall xLxa, \exists xLax \vdash \neg Laa$$

$$\begin{array}{c}
 \neg \forall xLxa \quad \checkmark \\
 \exists xLax \quad \checkmark \\
 \neg \neg Laa \quad \checkmark \\
 Laa \\
 \exists x \neg Lxa \quad \checkmark \\
 \neg Lba \\
 Lac
 \end{array}$$

The branch does not close. The argument is not valid.

The counterexample starts with the domain, as always in predicate logic:

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

How are we providing the extensions of the two-place predicates, however? We cannot just put all the names in it. If we did, we would neglect the important difference it often makes, in which argument place a name occurs. We thus help ourselves to so-called *ordered tuples*.

Ordered tuples are like sets, except that the *order* in which the members are written down is significant. Instead of curly set parentheses we use pointy parentheses to mark the difference. The set $\{a, b\}$ is the same as the set $\{b, a\}$, the ordered tuple $\langle a, b \rangle$, however, is different from $\langle b, a \rangle$.

In order to determine how long a tuple is, we also speak of n -tuples, where n is the number of places in the tuple. $\langle a, c \rangle$ is a two-tuple, for example. We also call ordered two-tuples *ordered pairs*. $\langle a, c, d \rangle$ is a three-tuple, or an *ordered triple*; $\langle a, c, d, c \rangle$ is a four-tuple, or an *ordered quadruple*; etc.

The idea now is to say that a relation L holding between a and b is like L being true of the ordered pair $\langle a, b \rangle$. In this way, then, we can easily determine the extension of many-place predicates.

To go back to our tree for ' $\neg \forall x Lxa, \exists x Lax \vdash \neg Laa$ ':

$$\begin{array}{l}
 \neg \forall x Lxa \quad \checkmark \\
 \exists x Lax \quad \checkmark \\
 \neg \neg Laa \quad \checkmark \\
 Laa \\
 \exists x \neg Lxa \quad \checkmark \\
 \neg Lba \\
 Lac
 \end{array}$$

The branch does not close. The argument is not valid.

Counterexample:

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

$$I(L) = \{\langle a, c \rangle, \langle a, a \rangle\}$$

An extension of a many-place predicate is thus just as easily read off an open branch of a tree as an extension of a one-place predicate.

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

$$(1) \exists x Lxm \vdash \forall x Lxm$$

$$(2) \forall y Lmy \vdash \exists y Lym$$

$$(3) \forall x(Fx \vee Hxa), \neg Haa \vdash \forall x(Fx \rightarrow Hxa)$$

$$(4) \forall x(Fx \rightarrow Gbx), \forall z(Gbz \rightarrow Hz), Fa \vdash Ha$$