

PY1003

Lecture 18**Number, Definite Descriptions,
and Function****Recap: Number**

There are at least two things that are F :

$$\exists x \exists y ((Fx \wedge Fy) \wedge \neg x=y)$$

There are at most two things that are F :

$$\forall x \forall y \forall z (((Fx \wedge Fy) \wedge Fz) \rightarrow ((x=z \vee x=y) \vee x=z))$$

There are exactly two things that are F :

$$\exists x \exists y [((Fx \wedge Fy) \wedge \neg x=y) \wedge \forall z (Fz \rightarrow (z=x \vee z=y))]$$

Recap: Definite Descriptions

The present King of France is bald:

$$\exists x [(Kx \wedge \forall y (Ky \rightarrow y=x)) \wedge Bx]$$

The present King of France is not bald:

$$\exists x [(Kx \wedge \forall y (Ky \rightarrow y=x)) \wedge \neg Bx]$$

Exercise:

Translate the following argument into the language of predicate logic. Provide a translation key. Determine, using the tree method, whether the argument is valid:

The Prime Minister of the U.K. is funny.
Therefore, there is at most one Prime Minister.

Only

Using analogous constructions, we can also express in the formal language of predicate logic with identity like the following one:

Tony loves only George.

Key: t : Tony
 g : George
 Rxy : x loves y

$Rtg \wedge \forall x(Rtx \rightarrow x = g)$

Again, there is a more elegant way of formalising the sentence:

$\forall x(Rtx \leftrightarrow x = g)$

Exercise:

Proof that $Rtg \wedge \forall x(Rtx \rightarrow x = g) \vdash \forall x(Rtx \leftrightarrow x = g)$, using the tree method. (The other direction is left as a take-away exercise.)

Note: We translated sentences like ‘Only mammals are whales’ as ‘ $\forall x(Wx \rightarrow Mx)$ ’ (Wx : x is a whale; Mx : x is a mammal) without assuming that $\exists x Wx$. In the case above, however, we *do* assume, that Tony indeed loves George – and not merely at most George.

Functions

Let’s say that Amy’s mum is cool. We can express this sentence (with the key: a : Amy, Mxy : x is y ’s Mum, Cx : x is cool) in this way:

$$\exists x[(Mxa \wedge \forall y(My a \rightarrow y=x)) \wedge Cx]$$

However, everyone has one and only one (biological) mother. It seems unnecessarily complicated to use the definite description, given that we have as a safe background assumption that everyone has exactly one mother.

We can use *functions* in these cases. A function always assigns exactly one object to an object that it applies to. We can write

$$f(a)$$

for ‘the father of Amy’. ‘ $f(a)$ ’ is an expression that refers to exactly one object – Amy’s father. We can thus use these expressions in the same way we use names, but not in the same way we use sentences. We cannot write ‘ $f(a)$ ’ alone as the antecedent of a conditional, for example. Only full sentences can occur in such a position; ‘the father of Amy’, however, is not a sentence.

We call expression like ‘ $f(a)$ ’ *terms*. Names, like ‘ a ’ are also terms.

We can thus generalise and say that one- or many-place predicates take *terms* as argument places, i.e. names (as we had it before) and function-expressions like ' $f(a)$ '.

So, we can, e.g., express

Amy's father is cool.

as:

$$Cf(a)$$

We can also make identity statements using both kinds of terms. So,

Phil is (identical to) Amy's father.

can be expressed as:

$$p = f(a)$$

It is **very important** that functions can only be used when *in general* the expression refers to *exactly one* object (or person). 'The brother of x ', for example, cannot be expressed using a function. Some folk have exactly one brother, but others have more than one, and yet others have no brother. Marcus, for example, has no brother. 'The brother of Marcus', therefore, does not refer – just like 'the present King of France'. Terms must always refer, however. That goes for names as well as function expressions. 'The brother of x ' must there be expressed using a definite description.

Note that functions do not require that everything the function applies to gets a *different* object assigned. E.g.: Amy and Jessie are sisters, so they have the same father. In other words, Amy's father is (identical to) Jessie's father; formally:

$$f(a)=f(j)$$

Function letters cannot only be applied to names, but to terms in general.

$$f(f(a))$$

for example, refers to the father of Amy's father – Amy's paternal grandfather. If we use ' $m(x)$ ' as the function 'the mother of x ', then

$$f(m(a))$$

refers to the father of Amy's mother – Amy's maternal grandfather, and

$$m(f(a))$$

refers to the mother of Amy's father – Amy's paternal grandmother. In this way,

Amy's father's mother is cool.

can be translated as:

$$Cm(f(a))$$

Exercises:

Translate the following, using function expressions:

- (1) Lisa likes John's twin brother.
- (2) Tom's mother sees the father of Randy's father.
- (3) Every boy loves his mother.
- (4) Anna is a mother.
- (5) Every mother is a father's mother.

Using the key given below, what do the formal sentences (6) – (9) express?

Key: m : Marcus domain : people
 $n(x)$: x 's next of kin
 Fx : x is funny
 Hxy : x hates y

- (6) $Fn(m)$
- (7) $\forall y \exists x x = n(y)$
- (8) $\exists x \exists y Hn(n(y))x$
- (9) $\forall x \forall y (Fy \rightarrow Hxn(y))$

A note on functions in trees (in case you wonder):

In principle the tree rules remain unchanged. Introducing functions into our logic means, however, that now all rules use *terms* in general where before only names were used.