Philosophy 240: Symbolic Logic
Fall 2010
Mondays, Wednesdays, Fridays: 9am - 9:50am

Hamilton College
Russell Marcus
rmarcus1@hamilton.edu

Class 23: October 20
Translation, Predicate Logic II (§8.1)

I. Characterizing a variety of predicate logics

We are starting our study of predicate logic by considering the simplest version of the language: Monadic Predicate Logic.
Predicate logic is monadic if the predicates only take one object.
When predicates take more than one object, we call the predicates relational, and we call the resulting logic Full First-Order Predicate Logic.
Consider:

Andrés loves Beatriz

We could regiment it in monadic predicate logic:

La

In that case, 'a' stands for Andrés, and 'L' stands for the property of loving Beatriz.
But, if we want to use 'L' to stand for the property of loving, it will have to take two objects: the lover and the lovee.
Thus, we could introduce a relational predicate, 'Lxy', which means that x loves y:

Lab

Relational predicates will allow us greater generality in our translation, just as translating negative statements using negations did.
We will look to reveal as much logical structure as we can.
But, relational predicates will come later, as an extension of Monadic Predicate Logic.

We will extend the language in other ways too, including a specific predicate for identity, and adding functors and second-order quantifiers.
A predicate logic is first-order when variables only appear in the object position.
In second-order logic, we add predicate variables.

When we studied propositional logic, we dove right into the full version of the language, and its formation rules, and then used the language in a derivation system.
For predicate logic, we enter the pool slowly.
Each time we extend the logic, we will generate a slightly new language, with slightly new formation rules.

Let's name each of the languages that we have seen or will see this term:

PL: Propositional Logic
M: Monadic (First-Order) Predicate Logic
F: Full (First-Order) Predicate Logic
FF: Full (First-Order) Predicate Logic with functors
S: Second-Order Predicate Logic

Another difference between our study of propositional logic and our study of predicate logic is that we will see the difference between a logical language and a system of deduction.
In propositional logic, we used one language, and one set of inference rules.
Technically, logicians separate a language from a deductive system.
We can use the same language in different deductive systems, and we can use the same deductive system with different languages.
We will use **M** and **F** with the same deductive system.
Then, we will introduce a new deductive system using the language **F**, adding new rules covering a special identity predicate.
It is typical to name both the deductive systems and the languages, but I think it will add too much confusion to do so in this course.

II. Formation rules for M

In constructing a formal language, we first specify the language, and then rules for wffs.
I will also introduce a few definitions.
We have already seen much of the content of this section, informally, but I wanted to be precise, and Hurley does not present formation rules.
It will be easier to compare the various language we will study if the vocabulary and formation rules are presented clearly.

Vocabulary of M

Capital letters A...Z, used as one-place predicates
Lower case letters
    a, b, c,...u are used as constants.
    v, w, x, y, z are used as variables.
Five connectives: ~, •, ∨, ⊃ ≡
Quantifier: ∃
Punctuation: (), [], {}

Constants and variables are called **terms**.
Some first-order systems allow propositional variables, like the ones we used in **PL**.
We could allow capital letters with no constants or variables directly following them to be propositional variables, as if they were zero-place predicates.
But we will not use them.

In order to use quantifers properly, one has to be sensitive to their scope.
Compare:

1. (x)(Px ⊃ Qx) Every P is Q
2. (x)Px ⊃ Qx If everything is P, then x is Q

The difference between these two expressions is the scope of the quantifier.
We have already tacitly seen something about scope in using negations.

The **scope of a negation** (in propositional logic) is whatever directly follows the negation symbol.

If what follows the negation is a single propositional variable, then the scope of the negation is just that propositional variable.
If what follows the negation is another negation symbol, then the scope of the first negation symbol is the scope of the second negation symbol plus that second negation symbol.
If what follows the negation is a bracket, then the entire formula which occurs between the opening and closing of that bracket is in the scope of the negation.

Similarly, the **scope of the quantifier** is whatever formula immediately follows the quantifier.

If what follows the quantifier is a bracket, then any formulas that occur until that bracket is closed are in the scope of the quantifier.
If what follows the quantifier is a negation, then every formula in the scope of the negation is in the scope of the quantifier.

Quantifiers bind every instance of their variable in their scope.
A **bound variable** is attached to the quantifier which binds it.

In 1, the 'x' in 'Qx' is bound.
In 2, the 'x' in 'Qx' is not bound.

An unbound variable is called a **free variable**.
Binding will become extremely important once we get to derivations, especially once we get to relational predicates.

The following open sentences contain both bound and free variables:

3. (x)Px ∨ Qx
4. (∃x)(Px ∨ Qy)

In 3, 'Qx' is not in the scope of the quantifier, so is unbound.
In 4, 'Qy' is in the scope of the quantifier, but 'y' is not the quantifier variable, so is unbound.

Formation rules for wffs of M

1. A predicate (capital letter) followed by a constant or variable (lower-case letter) is a wff.
2. If α is a wff, so are
   (∃x)α, (∃y)α, (∃z)α, (∃w)α, (∃v)α
   (x)α, (y)α, (z)α, (w)α, (v)α
3. If α is a wff, so is ~α.
4. If α and β are wffs, then so are:
   (α · β)
   (α ∨ β)
   (α ⊃ β)
   (α ≡ β)
   By convention, you may drop the outermost brackets.
5. These are the only ways to make wffs.

A few more terms:
A wff constructed only using rule 1 is called an **atomic formula**.
Here are some atomic formulae:

Pa
Qt
Ax

A wff that is part of another wff is called a **subformula**.
In '(Pa • Qb) ⊃ (∃x)Rx', the following are all subformulae:

Pa
Qb
Rx
(∃x)Rx
Pa • Qb

Wffs that contain at least one unbound variable are called **open sentences**.
Here are a couple of open sentences:

Ax
(x)(Px ⊃ Qx) ⊃ Rz

If a wff has no free variables, it is a **closed sentence**, and expresses a **statement**, or **proposition**.
Here are some closed sentences:

(y)[(Py • Qy) ⊃ (Ra ∨ Sa)]
(∃x)(Px • Qx) ∨ (y)(Ay ⊃ By)

Translations into **M** should yield closed sentences.
Quantifiers and connectives are called **operator**s, or logical operators.
Note that atomic formulas lack operators.
The last operator added according to the formation rules is called the **main operator**.

III. On extending the language

We are using only a small, finite stock of terms and quantifiers.
It is customary to use a larger stock, in fact an infinite stock.
To generate an indefinite number of terms and quantifiers, we could use the indexing functions of subscripts and superscripts.
We introduce arabic numberals, say, into the language.
Then, we index each constant and variable:

a_1, a_2, a_3...
x_1, x_2, x_3...

We can create an indefinite number of quantifiers by using the indexed variables.
More austere languages avoid introducing numbers, and just use 's.

a', a", a'", a""...
x', x", x'", x""....

Both of these techniques become ugly quickly.
Since we are only going to need a few variables and constants, we can use a cleaner syntax.

IV. Exercises. Translate each sentence into predicate logic.

1. All mice are purple. (Mx, Px)
2. No mice are purple.
3. Some mice are purple.
4. Some mice are not purple.
5. Snakes are reptiles. (Sx, Rx)
6. Snakes are not all poisonous. (Sx, Px)
7. Children are present. (Cx, Px)
8. Executives all have secretaries. (Ex, Sx)
9. Only executives have secretaries.
10. All that glitters is not gold. (Gx, Ax)
11. Nothing in the house escaped destruction. (Hx, Ex)
12. Blessed is he that considers the poor. (Bx, Cx)
13. Some students are intelligent and hard working. (Sx, Ix, Hx)
14. He that hates dissembles with his lips, and lays up deceit within him. (Hx, Dx, Lx)
15. Everything enjoyable is either illegal, immoral, or fattening. (Ex, Lx, Mx, Fx)
16. Some medicines are dangerous if taken in excessive amounts. (Mx, Dx, Tx)
17. Some medicines are dangerous only if taken in excessive amounts.
18. Victorian houses are attractive (Vx, Hx, Ax)
19. Slow children are at play. (Sx, Cx, Px)
20. Any horse that is gentle has been well-trained. (Hx, Gx, Wx)
21. Only well-trained horses are gentle.
22. Only gentle horses have been well-trained.
23. A knowledgeable, inexpensive mechanic is hard to find. (Kx, Ex, Mx, Hx)
24. Dogs and cats chase birds and squirrels. (Dx, Cx, Bx, Sx)

25. If all survivors are women, then some women are fortunate. (Sx, Wx, Fx)
26. Some, but not all, of us got away. (Ux, Gx)
27. If all ripe bananas are yellow, then some yellow things are ripe. (Rx, Bx, Yx)
28. If any employees are lazy and some positions have no future, then some employees will not be successful. (Ex, Lx, Px, Fx, Sx)
29. No coat is waterproof unless it has been specially treated. (Cx, Wx, Sx)
30. A professor is a good lecturer if and only if he is both well-informed and entertaining. (Px, Gx, Wx, Ex)

V. Solutions

1. $(x)(Mx \supset Px)$
2. $(x)(Mx \supset \sim Px)$
3. $(\exists x)(Mx \cdot Px)$
4. $(\exists x)(Mx \cdot \sim Px)$
5. $(x)(Sx \supset Rx)$
6. $\sim(x)(Sx \supset Px)$ or $(\exists x)(Sx \cdot \sim Px)$
7. $(\exists x)(Cx \cdot Px)$
8. $(x)(Ex \supset Sx)$
9. $(x)(Sx \supset Ex)$
10. $\sim(x)(Gx \supset Ax)$
11. $(x)(Hx \supset \sim Ex)$
12. $(x)(Cx \supset Bx)$
13. $(\exists x)[Sx \cdot (Ix \cdot Hx)]$
14. $(x)[Hx \supset (Dx \cdot Lx)]$
15. $(x)\{Ex \supset [(\sim Lx \vee \sim Mx) \vee Fx]$
16. $(\exists x)[Mx \cdot (Tx \supset Dx)]$
17. $(\exists x)[Mx \cdot (Dx \supset Tx)]$
18. $(x)[(Hx \cdot Vx) \supset Ax]$
19. $(\exists x)[(Cx \cdot Sx) \cdot Px]$
20. $(x)[(Hx \cdot Gx) \supset Wx]$
21. $(x)[(Hx \cdot Gx) \supset Wx]$
22. $(x)[(Hx \cdot Wx) \supset Gx]$
23. $(x)\{[(Kx \cdot \sim Ex) \cdot Mx] \supset Hx\}$
24. $(x)[(Dx \vee Cx) \supset (Bx \cdot Sx)]$
25. $(x)(Sx \supset Wx) \supset (\exists x)(Wx \cdot Fx)$
26. $(\exists x)(Ux \cdot Gx) \cdot \sim(x)(Ux \supset Gx)$
27. $(x)[(Bx \cdot Rx) \supset Yx] \supset (\exists x)(Yx \cdot Rx)$
28. $[(\exists x)(Ex \cdot Lx) \cdot (\exists x)(Px \cdot \sim Fx)] \supset (\exists x)(Ex \cdot \sim Sx)$
29. $(x)[Cx \supset (\sim Wx \vee Sx)]$ or $(x)[Cx \supset (\sim Sx \supset \sim Wx)]$ or $(x)[(Cx \cdot Wx) \supset Sx]$ or $\sim(\exists x)(Cx \cdot Wx \cdot \sim Sx)$
30. $(x)\{Px \supset [Gx \equiv (Wx \cdot Ex)]\}$