AIMA Chapter 8 lecture notes, October 16 and 18, 2006
Dr. Gene B. Chase

Don’t worry that we’ve rushed over resolution as a theorem proving method. We’ll meet it again in Chapter 9.

Read §§8.1–8.3 for Wednesday.

Let’s create a logic of which statement logic is a special case: predicate logic. It consists of

	Math	Prolog	Real world	Language
A. Terms			**things**	**noun phrases**
1. constants	a, b, c	`joe, yesterday` `mary`		**nouns**

Your text also calls these “ground terms.”

	Math	Prolog	Real world	Language
2. functions	father(x) x^2+5	[none]	[later]	[later]
B. Variables	x, y, z	`X, Y, Z`	–	**pronouns**
C. Predicates	p(x), q(x,y)	`happy(X),` `loves(X, Y)`	**attributes, relations**	**adjectives, verbs**
allow	$\wedge$ $\vee$ $\neg$ $\Rightarrow$	`, ; not :-`	—	**conjunctions**
D. Quantifiers				
there exists	$\exists x$	[later]		**some, an**
for all	$\forall x$	[later]		**each, every, any**

In a finite domain, both would be unnecessary, because
$\exists$ would just be $\vee$ and $\forall$ would just be $\wedge$. Example: Wumpus world.

We’ll use both even though one can be defined in terms of the other.
$\neg\exists x\neg p(x) = \forall x\ p(x)$ “If you can’t find something to falsify p(x), then p(x) is true for all x.”
$\neg\forall x\neg p(x) = \exists x\ p(x)$ “If all x’s do not falsify p(x), then p(x) is true for one x.”

You can see echos of DeMorgan’s laws in these rules [p. 252]

	Math	Prolog	Real world	Language
E. Equality				
	=	[none]	identity	is
joe = father(mary)			[not similarity or congruence]	joe is mary’s father

F. Two new logical axioms

Existential generalization

$$\frac{p(c)}{\exists x\, p(x)}$$

Universal instantiation

$$\frac{\forall x\, p(x)}{p(c)}$$

[where c is a constant not seen yet]

From now on, when we write ⊢ we mean "in predicate calculus." There will of course be additional domain-specific axioms [pp. 254–258] to write on the left hand side for—

- natural numbers • the Wumpus world. • kinship • electronic circuits [§8.4]

G. To talk about truth we must have a model of predicate logic.

Constants:	Chosen from a set of things, the domain of the constant.
Functions:	Mathematical functions on those sets.
Predicates:	Mathematical relations (sets of ordered pairs) on those sets.

Practice, English to predicate calculus and vice versa. The tricky ones to get right are those involving quantifiers:

All men are mortal.	$\forall x$ man(x) ⇒ mortal(x).
Some leaders are charismatic.	$\exists x$ leader(x) ∧ charismatic(x).

H. Is it still sound and complete? Yes. Everything that's true is provable, and everything that's provable is true. [Aside: Even adding axioms for < and +, predicate calculus on the domain of natural numbers is complete. But add also * and it's not. Result for + due to M. Pressburger, 1929; for * due to K. Gödel, 1931.]

I. Practice.

1. Ambiguity of English.

   pp. 251–252 Everyone loves someone.
   $\forall x \exists y$ loves(x,y)
   $\exists x \forall y$ loves(x,y)

2. As much as there is time on Wednesday, we'll work on Exercises 8.3, 8.4, 8.6, 8.7, 8.11a

Sample	8.6 d. Multiple ontologies permit multiple answers
Solution 1	domain of Language: {greek, french}
	domain of Score: [0,1, 2, ..., 100]
	function: `highestScore (Language)` with values as `Score`
	`higherThan (Score, Score)` [possibly equals]
Answer 1	`higherThan(highestScore(greek), highestScore(french))`
Solution 2	same domains; now try to eliminate highestScore(Language) using higherThan
	but add a function: `score(Language, Test)` with values as Score
English 2	There is a score in Greek which is higher than all other scores in Greek,
	and there is a score in French which is higher than any other score in French,
	and the first score is higher than the second.
Answer 2	∃ Test1 ∃ Test2 ∀ Test3 ∀ Test4
	higherThan(score(greek, Test1), score(greek, Test3))
	∧ higherThan(score(french, Test2), score(french, Test4))
	∧ higherThan(score(greek, Test1), score(french, Score2))

Of course you can use ≥ instead of higherThan.
Is your ontology in Exercise 8.6a–c consistent with the above? If not, modify the above to make it so. Perhaps you'll use `take(Student, Course, Semester)`, and `pass(Student, Course, Semester)`.