MAU22C00 Lecture 8

John Stalker

Trinity College Dublin

Lessons for writing informal proofs

Try to make things easy on your reader. If it's not obvious why you're introducing a hypothesis then add some words to indicated what you're doing, e.g. "A is either finite or infinite. Suppose A is finite. Then ... Suppose A is infinite. Then ... In either case we see that ..." is better than just jumping to "Suppose A is finite", which could be setting up a proof by contradiction.

Try to prove things in the broadest scope in which they are true, not the scope where you first need them, where some unnecessary hypotheses may have been introduced. Usually this means going back and inserting a proof before those hypotheses were introduced. Proofs are sometimes easier to write out of sequence!

Substitution is dangerous! The rules governing it are not obvious. You need to learn them.

First order logic

First order logic "extends" zeroeth order logic.

It introduces quantifiers: the universal quantifier $\forall$, read "for all", and the existential quantifier $\exists$, read "for some".

It also introduces predicates, variables and parameters, which we'll discuss next.

It's not a true extension because we also remove something: Boolean variables.

First order logic is meant as a base for more interesting systems, e.g. elementary arithmetic, set theory, etc.

Predicates, variables and parameters

Variables in first order logic are stand-ins for variables in whatever system we'll be using first order logic as a base for, e.g. natural number variables in elementary arithmetic, set variables in set theory, etc.

Parameters are stand-ins for expressions of the same type as the variables, e.g. natural number expressions in elementary arithmetic, set expressions in set theory, etc.

Predicates are stand-ins for Boolean expressions. These can depend on variables or parameters.

For example in elementary arithmetic we might replace a variable from first order logic with a numerical variable like x, replace a parameter by a numerical expression like $y + z$ and replace a predicate by something like "is prime", or rather by its translation into the language of elementary arithmetic.

For example in set theory we might replace a variable from first order logic with a set variable like A, replace a parameter by a set expression like $B \bigcap C$ and replace a predicate by something like "is finite", or rather by its translation into the language of set theory.

The language of first order logic

```
statement      : expression ;
expression     : atomic_expression | ( expression binop expression )
               | [ expression binop expression ]
               | { expression binop expression } | ( ¬ expression )
               | [ ¬ expression ] | { ¬ expression }
               | ( quantifier variable . expression )
               | [ quantifier variable . expression ]
               | { quantifier variable . expression } ;
atomic_expression : ( atom ) | [ atom ] | { atom } ;
atom           : predicate | atom variable | atom paramater ;
binop          : ∧ | ∨ | ⊃ | ⊼ | ⊻ | ≡ | ≢ | ⊂ ;
quantifier     : ∀ | ∃ ;
predicate      : pred_letter | predicate ! ;
pred_letter    : f | g | h | i | j ;
parameter      : param_letter | parameter ! ;
param_letter   : a | b | c | d | e ;
variable       : var_letter | variable ! ;
var_letter     : v | w | x | y | z ;
```

Comments on the language

This grammar is unambiguous.

I'm still keeping three sets of brackets for readability

We still have the option of using !'s to generate an unlimited number of variables, and still won't use it in examples.

Letters from the start of the alphabet are parameters, letters from the middle are predicates and letters from the end are variables.

The syntax for quantifiers is

```
( quantifier variable . expression )
```

Predicates go before their arguments, like fx or ga or hye, even though in applications they would often go after or in the midst of them, as in "x is prime", or "A is finite" or "A is a subset of $B \bigcup C$".

You can think of predicates as standing for Boolean functions of one or more arguments. This is technically wrong, but not very wrong.

Free and bound (occurences of) variables

An annoying, but vital, part of first order logic is the distinction between free and bound variables.

Consider the following statement from the arithmetic of natural numbers: "For all l there is an n such that $l = m + n$".

The role of m is different from the roles of l and n. It makes sense to pick a natural number m and ask whether the statement above is true or false.

Indeed we need to pick an m to ask that question, unless we're implicitly considering it as a statement about all m.

It does not make sense to pick an l or an n and ask whether the statement is true, since the equation $l = m + n$ needs to hold for all l and for some n, possibly depending on l.

We say that l and n are *bound*, while m is *free*.

Free and bound variables, continued.

In first order logic we have quantifiers but the role of expressions like $l = m + n$ is played by predicates. The analogue of the example before is that in

$$\{\forall x.[\exists z.(f\,x\,y\,z)]\}$$

the variable y is free but the variables x and z are bound.

A variable could be bound in one expression but free in a subexpression. x is free in $[\exists z.(f\,x\,y\,z)]$ but bound in $\{\forall x.[\exists z.(f\,x\,y\,z)]\}$.

Even in the same expression one occurence of a variable could be free while another is bound. This is considered bad style, but isn't forbidden.

The rules for which occurences of which variables are bound in which expressions aren't part of the grammar but are defined in terms of it.

Rules for free and bound variables

- In an atomic expression all occurences of all variables are free.
- Combining expressions with Boolean operators doesn't change the status of any variables.
- Quantifiers convert all free occurences of the variable following the quantifier to bound occurences, but leave the status of other variables unchanged.

Example: $(fxyz)$ is atomic, so x, y and z are all free in it. $[\exists z.(fxyz)]$ has a quantifier in front of a z so z is bound in this larger expression but x and y remain free. $\{\forall x.[\exists z.(fxyz)]\}$ has a quantifier in front of x so x is bound in it. z remains bound and y remains free.

Order of quantifiers

Order matters.

For which m is the statement "For all l there is an n such that $l = m + n$" true?

Start from the inside and work your way out. For which l and m is "there is an n such that $l = m + n$" true?

I said we're working with natural numbers, so this l must be non-negative. We need, $m \leq l$, and that's also sufficient.

So "for all l there is an n such that $l = m + n$" is the same as "for all l we have $m \leq l$", i.e. m is less than or equal to every natural number. This is true for exactly natural number m, namely 0.

What about "there is an n such that for all l we have $l = m + n$"?

Start from inside again. For which m and n is it true that "for all l we have $l = m + n$"?

None! There is at most one l such that $l = m + n$.

So "there is an n such that for all l we have $l = m + n$" is not true for any m.

Order of quantifiers, continued.

"For all l there is an n such that $l = m + n$" is true for $m = 0$.

"There is an n such that for all l we have $l = m + n$" is not true for any m.

These are definitely not the same statement.

What about "there is an n such that $l = m + n$ for all l"?

This could mean "(there is an n such that $l = m + n$) for all l" or "there is an n such that ($l = m + n$ for all l)".

If you put some quantifiers at the start and some at the end people will have to guess what you mean. If you put them all at the start (or all at the end) then the meaning is unambiguous.

Exceptions: If you have only one quantifier it doesn't matter where it goes. The order of quantifiers *of the same type* doesn't matter.

Interpretation

Variables and parameters are thought of as belonging to a "domain", which may be a set.

If it is a set then predicates can be thought of as Boolean functions (of some number of arguments) on that set.

$\forall$ followed by a variable and an expression means the expression evaluates to true whenever *any* element of the domain is substituted for all *free* occurences of that variable in that expression.

$\exists$ followed by a variable and an expression means the expression evaluates to true when *some* element of the domain is substituted for all free occurences of that variable in that expression.

This is one of many contexts where free and bound occurences behave differently.

Classical first order logic assumes the domain is non-empty!