C140 Logic

Alessandra Russo
{a.russo@imperial.ac.uk}
Department of Computing Imperial College London

Notes based on Ian Hodkinson's material
Thanks to Krysia Broda for additional material

Academic year 2017 - 2018

Table of contents I

Layout

- **10 lectures**, and **5 tutorials**
  - Predicate logic
    Note: predicate logic is also known as first-order logic or sometime as "classical logic".

- **PMT weekly tutorials**. Tutorial 5 will be run in the lab for you to try the Pandora program on natural deduction.

- **Xmas test**: 25-minute logic question near the end of term

Relevance with other courses

Logic

Discrete Mathematics	Reasoning about Programs	Databases I		1st year courses
Models of Computation	Introduction to Artificial Intelligence	Introduction to Prolog	......	2nd year courses
Advanced Databases	Systems Verification	Logic-based Learning	......	3rd year courses
Modal Logic	Knowledge Representation	Separation Logic	......	4th year courses

You may need Logic to answer exam questions in these courses.

Classical First-Order Predicate Logic

Outline

This is a powerful extension of propositional logic. It is the most important logic of all.

In this second part of the course we will:

- explain predicate logic syntax and semantics carefully
- do English – predicate logic translation, and see examples from computing (pre- and post-conditions)
- generalise arguments and validity from propositional logic to predicate logic
- consider ways of establishing validity in predicate logic:
  - truth tables — they don't work
  - direct argument — very useful
  - equivalences — also useful
  - natural deduction (sorry).

Outline

This is a powerful extension of propositional logic. It is the most important logic of all.

In this second part of the course we will:

- explain predicate logic syntax and semantics carefully
- do English – predicate logic translation, and see examples from computing (pre- and post-conditions)
- generalise arguments and validity from propositional logic to predicate logic
- consider ways of establishing validity in predicate logic:
    - truth tables — they don't work
    - direct argument — very useful
    - equivalences — also useful
    - natural deduction (sorry).

Outline

This is a powerful extension of propositional logic. It is the most important logic of all.

In this second part of the course we will:

- explain predicate logic syntax and semantics carefully
- do English – predicate logic translation, and see examples from computing (pre- and post-conditions)
- generalise arguments and validity from propositional logic to predicate logic
- consider ways of establishing validity in predicate logic:
  - truth tables — they don't work
  - direct argument — very useful
  - equivalences — also useful
  - natural deduction (sorry).

Outline

This is a powerful extension of propositional logic. It is the most important logic of all.

In this second part of the course we will:

- explain predicate logic syntax and semantics carefully
- do English – predicate logic translation, and see examples from computing (pre- and post-conditions)
- generalise arguments and validity from propositional logic to predicate logic
- consider ways of establishing validity in predicate logic:
  - truth tables — they don't work
  - direct argument — very useful
  - equivalences — also useful
  - natural deduction (sorry).

Why predicate logic?

Propositional logic is quite nice, but not very expressive.

Statements like

- the list is ordered
- every worker has a boss
- there is someone worse off than you

need something more than propositional logic to express.

Propositional logic cannot express arguments like this one of De Morgan:

- A horse is an animal.
- Therefore, the head of a horse is the head of an animal.

Why predicate logic?

Propositional logic is quite nice, but not very expressive.

Statements like

- the list is ordered
- every worker has a boss
- there is someone worse off than you

need something more than propositional logic to express.

Propositional logic cannot express arguments like this one of De Morgan:

- A horse is an animal.
- Therefore, the head of a horse is the head of an animal.

Predicate logic in a nutshell

- Syntactically, there are 5 new features:
  1. Atomic formulas become Structured (i.e. they take parameters). Eg. `sister(me,you)`
  2. Quantifiers (*for all*, *there exists*).
  3. Variables $x, y, z, \ldots$.
  4. Equality $=$ is included
  5. Function symbols. (abstract versions of arithmetical $+, -, \times, \surd$), and sorted (typed) variables.

- Semantically, the notion of *situation* is more complex than for propositional logic. We have to give meaning to the predicates, to the variables, and to the quantifiers.

1.1 Syntax

Splitting the atoms - new atomic formulas

Up to now, we have regarded phrases such as the computer is a PC and Frank bought grapes as atomic, without internal structure. Now we look inside them.

We regard "being a PC" as a property or attribute that a computer (and other things) may or may not have.

So we introduce:

- Relation symbols (or predicate symbols)
  - `PC`. It takes 1 argument — we say it is *unary* or its 'arity' is 1.
  - `bought`. It takes 2 arguments — we say it is *binary*, or its arity is 2.
- Constants, to name objects
  - `Heron`, `Frank`, `grapes`, . . .,

Then `PC(Heron)` and `bought(Frank,grapes)` are examples of new atomic formulas.

Splitting the atoms - new atomic formulas

Up to now, we have regarded phrases such as the computer is a PC and Frank bought grapes as atomic, without internal structure.
Now we look inside them.

We regard "being a PC" as a property or attribute that a computer (and other things) may or may not have.

So we introduce:

- Relation symbols (or predicate symbols)
  - `PC`. It takes 1 argument — we say it is *unary* or its 'arity' is 1.
  - `bought`. It takes 2 arguments — we say it is *binary*, or its arity is 2.
- Constants, to name objects
  - `Heron`, `Frank`, `grapes`, ...,

Then `PC(Heron)` and `bought(Frank,grapes)` are examples of new atomic formulas.

Splitting the atoms - new atomic formulas

Up to now, we have regarded phrases such as the computer is a PC and Frank bought grapes as atomic, without internal structure. Now we look inside them.

We regard "being a PC" as a property or attribute that a computer (and other things) may or may not have.

So we introduce:

- Relation symbols (or predicate symbols)
  - `PC`. It takes 1 argument — we say it is *unary* or its 'arity' is 1.
  - `bought`. It takes 2 arguments — we say it is *binary*, or its arity is 2.
- Constants, to name objects
  - `Heron`, `Frank`, `grapes`, . . .,

Then `PC(Heron)` and `bought(Frank,grapes)` are examples of new atomic formulas.

Splitting the atoms - new atomic formulas

Up to now, we have regarded phrases such as the computer is a PC and Frank bought grapes as atomic, without internal structure. Now we look inside them.

We regard "being a PC" as a property or attribute that a computer (and other things) may or may not have.

So we introduce:

- Relation symbols (or predicate symbols)
  - `PC`. It takes 1 argument — we say it is *unary* or its 'arity' is 1.
  - `bought`. It takes 2 arguments — we say it is *binary*, or its arity is 2.
- Constants, to name objects
  - `Heron`, `Frank`, `grapes`, . . .,

Then `PC(Heron)` and `bought(Frank,grapes)` are examples of new atomic formulas.

Quantifiers

So what? You may think that writing

$$\texttt{bought}(\texttt{Frank}, \texttt{grapes})$$

is not much more exciting than what we did in propositional logic — writing

`Frank bought grapes.`

But predicate logic has machinery to vary the arguments to **`bought`**.

This allows us to express properties of the relation '**`bought`**'.

The machinery is called quantifiers.
(The word was introduced by De Morgan.)

What are quantifiers?

A quantifier specifies a quantity (of things that have some property).

Example 1.1

- All students work hard.
- Some students are asleep.
- Most lecturers are crazy.
- Eight out of ten cats prefer it.
- No one is worse off than me.
- At least six students are awake.
- There are infinitely many prime numbers.
- There are more PCs than there are Macs.

Quantifiers in predicate logic

There are just two:

- $\forall$ (or `(A)`): ‘for all’
- $\exists$ (or `(E)`): ‘there exists’ (or ‘some’)

Some other quantifiers can be expressed with these. (They can also express each other.)

But quantifiers like infinitely many and more than cannot be expressed in first-order logic in general. (They can in, e.g., second-order logic. And even first-order logic can sometimes express them in special cases.)

How do they work?

We’ve seen expressions like `Heron`, `Texel`, etc. These are constants, like π, or e. So, to express ‘All computers are PCs’ we need variables that range over all computers, not just Heron, Texel, etc.

Quantifiers in predicate logic

There are just two:

- $\forall$ (or `(A)`): 'for all'
- $\exists$ (or `(E)`): 'there exists' (or 'some')

Some other quantifiers can be expressed with these. (They can also express each other.)

But quantifiers like infinitely many and more than cannot be expressed in first-order logic in general. (They can in, e.g., second-order logic. And even first-order logic can sometimes express them in special cases.)

How do they work?

We've seen expressions like `Heron`, `Texel`, etc. These are constants, like π, or e. So, to express 'All computers are PCs' we need variables that range over all computers, not just Heron, Texel, etc.

Quantifiers in predicate logic

There are just two:

- $\forall$ (or `(A)`): 'for all'
- $\exists$ (or `(E)`): 'there exists' (or 'some')

Some other quantifiers can be expressed with these. (They can also express each other.)

But quantifiers like infinitely many and more than cannot be expressed in first-order logic in general. (They can in, e.g., second-order logic. And even first-order logic can sometimes express them in special cases.)

How do they work?

We've seen expressions like `Heron`, `Texel`, etc. These are constants, like π, or e. So, to express 'All computers are PCs' we need variables that range over all computers, not just Heron, Texel, etc.

Variables

We will use variables to do quantification. We fix an infinite collection (or 'set') V of variables: e.g., $x, y, z, u, v, w, x_0, x_1, x_2, \ldots$
Sometimes I write x or y to mean 'any variable'.
As well as formulas like $\mathtt{PC}(\mathtt{Heron})$, we'll write formulas like $\mathtt{PC}(\mathtt{x})$.

- Now, to say 'Everything is a PC', we'll write $\forall x\, \mathtt{PC}(\mathtt{x})$.
  This is read as: 'For all x, x is a PC'.
- 'Something is a PC', can be written $\exists x\, \mathtt{PC}(\mathtt{x})$.
  'There exists x such that x is a PC.'
- 'Frank bought a PC', can be written

  $$\exists x(\mathtt{PC}(\mathtt{x}) \wedge \mathtt{bought}(\mathtt{Frank}, x)).$$

  'There is an x such that x is a PC and Frank bought x.'
  Or: 'For some x, x is a PC and Frank bought x.'

We will now make all of this precise.

Signatures

Definition 1.2 (signature)

A signature is a collection (set) of constants, and relation symbols with specified arities.

Some call it a similarity type, or vocabulary, or (loosely) language.

It replaces the collection of propositional atoms we had in propositional logic.

We usually write L to denote a signature. We often write $c, d, \ldots$ for constants, and $P, Q, R, S, \ldots$ for relation symbols.
Later, we'll consider also function symbols.

Example of a simple signature

Which symbols we put in L depends on what we want to say.

For illustration, we'll use a handy signature L consisting of:

- constants `Frank`, `Susan`, `Tony`, `Heron`, `Texel`, `Clyde`, `Room-308`, and c
- unary relation symbols `PC`, `human`, `lecturer` (arity 1)
- a binary relation symbol `bought` (arity 2).

Warning: things in L are just symbols — syntax. They don't come with any meaning. To give them meaning, we'll need to work out (later) what a situation in predicate logic should be.

Terms

To write formulas, we'll need terms, to name objects.
Terms are not formulas. They will not be true or false.

Definition 1.3 (term)

Fix a signature L.

1. Any constant in L is an L-term.
2. Any variable is an L-term.
3. Nothing else is an L-term.

A closed term or (as computing people say) ground term is one that doesn't involve a variable.

Examples of terms

`Frank`, `Heron` (ground terms). x, y, x_{56} (not ground terms)
Later, we'll extend this notion to include function symbols as well.

Formulas of first-order logic

Definition 1.4 (formula)

Fix L as before.

1. If R is an n-ary relation symbol in L, and $t_1, \ldots, t_n$ are L-terms, then $R(t_1, \ldots, t_n)$ is an atomic L-formula.
2. If t, t' are L-terms then $t = t'$ is an atomic L-formula. (Equality — very useful!)
3. $\top, \bot$ are atomic L-formulas.
4. If A, B are L-formulas then so are $(\neg A)$, $(A \wedge B)$ $(A \vee B)$, $(A \to B)$, and $(A \leftrightarrow B)$.
5. If A is an L-formula and x a variable, then $(\forall x\, A)$ and $(\exists x\, A)$ are L-formulas.
6. Nothing else is an L-formula.

Examples of formulas

Binding conventions: as for propositional logic, plus: $\forall x, \exists x$ have same strength as $\neg$.

1. $\texttt{bought}(\texttt{Frank}, x)$
   We read this as: 'Frank bought x.'
2. $\exists x\ \texttt{bought}(\texttt{Frank}, x)$
   'Frank bought something.'
3. $\forall x(\texttt{lecturer}(x) \rightarrow \texttt{human}(x))$
   'Every lecturer is human.' [Important eg!]
4. $\forall x(\texttt{bought}(\texttt{Tony}, x) \rightarrow \texttt{PC}(x))$
   'Everything Tony bought is a PC,' or 'Tony bought only PCs'.

Formation trees and subformulas (see slides 19 - 21)), literals and clauses, etc., can be done much as before.

More examples

5. $\forall x(\texttt{bought}(\texttt{Tony}, x) \to \texttt{bought}(\texttt{Susan}, x))$
   'Susan bought everything that Tony bought.'
6. $\forall x\, \texttt{bought}(\texttt{Tony}, x) \to \forall x\, \texttt{bought}(\texttt{Susan}, x)$
   'If Tony bought everything, so did Susan.' Note the difference!
7. $\forall x \exists y\, \texttt{bought}(x, y)$
   'Everything bought something.'
8. $\exists y \forall x\, \texttt{bought}(x, y)$
   'There is something that everything bought.' Note the difference!
9. $\exists x \forall y\, \texttt{bought}(x, y)$
   'Something bought everything.'

You can see that predicate logic is rather powerful — and terse.

More examples

5. $\forall x(\texttt{bought}(\texttt{Tony}, x) \rightarrow \texttt{bought}(\texttt{Susan}, x))$
   'Susan bought everything that Tony bought.'
6. $\forall x\, \texttt{bought}(\texttt{Tony}, x) \rightarrow \forall x\, \texttt{bought}(\texttt{Susan}, x)$
   'If Tony bought everything, so did Susan.' Note the difference!
7. $\forall x \exists y\, \texttt{bought}(x, y)$
   'Everything bought something.'
8. $\exists y \forall x\, \texttt{bought}(x, y)$
   'There is something that everything bought.' Note the difference!
9. $\exists x \forall y\, \texttt{bought}(x, y)$
   'Something bought everything.'

You can see that predicate logic is rather powerful — and terse.

More examples

5. $\forall x(\texttt{bought}(\texttt{Tony}, x) \rightarrow \texttt{bought}(\texttt{Susan}, x))$
   'Susan bought everything that Tony bought.'
6. $\forall x\, \texttt{bought}(\texttt{Tony}, x) \rightarrow \forall x\, \texttt{bought}(\texttt{Susan}, x)$
   'If Tony bought everything, so did Susan.' Note the difference!
7. $\forall x \exists y\, \texttt{bought}(x, y)$
   'Everything bought something.'
8. $\exists y \forall x\, \texttt{bought}(x, y)$
   'There is something that everything bought.' Note the difference!
9. $\exists x \forall y\, \texttt{bought}(x, y)$
   'Something bought everything.'

You can see that predicate logic is rather powerful — and terse.

More examples

5. $\forall x(\texttt{bought}(\texttt{Tony}, x) \rightarrow \texttt{bought}(\texttt{Susan}, x))$
   'Susan bought everything that Tony bought.'
6. $\forall x\, \texttt{bought}(\texttt{Tony}, x) \rightarrow \forall x\, \texttt{bought}(\texttt{Susan}, x)$
   'If Tony bought everything, so did Susan.' Note the difference!
7. $\forall x \exists y\, \texttt{bought}(x, y)$
   'Everything bought something.'
8. $\exists y \forall x\, \texttt{bought}(x, y)$
   'There is something that everything bought.' Note the difference!
9. $\exists x \forall y\, \texttt{bought}(x, y)$
   'Something bought everything.'

You can see that predicate logic is rather powerful — and terse.

More examples

5. $\forall x(\texttt{bought}(\texttt{Tony}, x) \rightarrow \texttt{bought}(\texttt{Susan}, x))$
   'Susan bought everything that Tony bought.'
6. $\forall x\, \texttt{bought}(\texttt{Tony}, x) \rightarrow \forall x\, \texttt{bought}(\texttt{Susan}, x)$
   'If Tony bought everything, so did Susan.' Note the difference!
7. $\forall x \exists y\, \texttt{bought}(x, y)$
   'Everything bought something.'
8. $\exists y \forall x\, \texttt{bought}(x, y)$
   'There is something that everything bought.' Note the difference!
9. $\exists x \forall y\, \texttt{bought}(x, y)$
   'Something bought everything.'

You can see that predicate logic is rather powerful — and terse.