University of Malta
BSc(Hons.) IT Year I

Introduction to Logic and Logic Programming

Lecture 2

Atomic and Complex Sentences

- Function Symbols
- Functions and Predicates
- Complex Sentences
- Syntax and Semantics
- Truth Functional Connectives
- Backus Naur Form

Department of Computer Science and AI
1988/9

The First Order Language of Set Theory

The language has only two predicates, both binary and infix.

- The equality symbol $=$
- The set membership symbol $\in$

A sentence of the form $a = b$ is true if the constants a and b name one and the same thing.

A sentence of the form $a \in b$ is true if and only if b is the name of a set and a names an element of that set.

Exercise

Suppose a and b are names and the universe contains numbers, sets of numbers, sets of sets of numbers etc. Assign objects to a and b such that the following claims are as shown

sentence	**truth status**
$a \in a$	FALSE
$a \in b$	TRUE
$b \in a$	FALSE
$b \in b$	FALSE

Answer $a = 2$; $b = \{2,4,6\}$ Now suppose $c = 6$ and $d = \{2,7,\{2,4,6\}\}$

sentence	**truth status**
$a \in c$	?
$a \in d$	?
$b \in c$	?
$b \in d$	?
$c \in d$	?
$c \in b$	?

Function Symbols

Some first order languages include function symbols as well as names and predicates.

Function symbols allow us to form name-like terms from names and other name-like terms.

We can do this in English with phrases like

Max
Max's father
Max's father's wife
Max's father's wife's husband
The sum of 3 and 4

and in FOL with "terms" composed of a function symbol followed by an appropriate number of arguments

Max
father(Max)
wife(father(Max))
husband(wife(father(Max)))
sum(3,4) or 3+4

Warning

There is a big difference between function symbols and predicates, though they are often confused because they both take *terms* as arguments. A term is either a name or a name-like entity formed using a function.

The difference is:

- If you apply a predicate to the right number of terms, you get a sentence which makes a claim and has a truth status.
- If you apply a function to the right number of arguments, you get *another term*, which should refer to a **unique** object of some sort. This is why function symbols can be applied over and over again in a way that is impossible for predicates.

Predicates and Functions Together

Assuming father is a unary function and Hungry is a unary predicate, the following are well-formed *terms*

father(Mike)
father(father(Mike))
father(father(father(Mike)))

And the following are well formed atomic sentences

Hungry(Mike)
Hungry(father(Mike))
Hungry(father(father(Mike)))

However, the following are ill formed:

Hungry(Hungry(Mike))
father(Hungry(Mike))

The First Order Language of Arithmetic

1. **Description of Symbols**

Symbol	**Arity**	**Type**
$+$	2	function
$\times$	2	function
$=$	2	predicate
$>$	2	predicate
$<$	2	predicate

2. **Rules for the formation of Terms**

   - The names 0 and 1 are terms
   - If t1 and t2 are terms, then (t1 + t2) is a term and (t1 × t2) are terms

Example of an atomic sentence:

$$(1 \times 1) < (1 + 1)$$

General First Order Languages

- There is not one first order language, but many.
- A first order language is specified by fixing the names, predicates and function symbols (along with their respective arities).
- There *must* be at least one predicate symbol. Everything else is optional.
- Sometimes, the specification is already supplied. At other times (e.g. when translating from English into FOL), it has to be *designed*.
- Designing a first order language with just the right names, predicates and function symbols requires some skill

John gave Mary a kiss

Complex Sentences

Although individual members of the FOL differ considerably, what they share are the connectives and quantifiers that enable us to build complex sentences from atomic ones. In order to handle complex sentences, we need rules at two different levels:

- **Syntax**. Rules which tell us how to construct complex sentences using connectives and which distinguish well formed sentences from ill formed ones.
- **Semantics**. Rules which tell how to compute the meaning of complex sentences.

All the connectives are *truth functional*. This means that the truth value of a sentence built using these connectives depends *exclusively* on the truth values of the simpler sentences out of which it is built.

Truth Functionality

Note that in English the uses of some connectives appear to be truth functional, whilst the uses of others clearly are not.

(a) John ate steak and Mary ate steak
(b) John ate steak because Mary ate steak

What (a) claims is true if both John and Mary ate steak, whilst this is not sufficient for the truth of (b).

Clearly, the importance of truth-functional connectives is that we are *guaranteed* to be able to compute the truth value of a complex sentence by taking it apart according to the syntactic rules and computing the truth values of the parts.

This process, known as "truth recursion", terminates when we reach an atomic sentence whose truth or falsity is established by looking in the world.

Syntax of Propositional Calculus

Backus-Naur Form

Formula	::=	AtomicSent $\mid$ Complex
Complex	::=	(Formula Con Formula) $\mid$ $\neg$Formula
Con	::=	$\wedge \mid \vee \mid \rightarrow \mid \leftrightarrow$
AtomicSent	::=	p $\mid$ q $\mid$ r $\mid$ s

- "::=" means "rewrites as". "|" means "or" in the sense of two or more possible rewrites.
- We will sometimes use "&" instead of "$\wedge$" and "-" instead of "$\neg$" in what follows.
- For the propositional calculus we ignore the internal structure of atomic sentences. We can achieve this by assuming that our first order language only has predicates of arity 0.