University of Malta
BSc(Hons.) IT Year I

# Introduction to Logic and Logic Programming

## Lecture 1

## Introduction

- Course Information
- What is Logic?
- Arguments & Propositions & Atomic Sentences
- Overview

Department of Computer Science and AI
1988/9

# CSI-107: Preliminaries

| | |
|---|---|
| Photocopies | Konika |
| Web Site | • `http://www.cs.um.edu.mt/~mros/cs107`<br>• Course Information<br>• Online Notes (html)<br>• Lecture Slides (PostScript) |
| Software | • Prolog (download)<br>• Tarski's World |
| Email | `mros@cs.um.edu.mt` |
| Computers | PC Lab/James Mallia |
| Me | Office 416, CS<br>tel. 3290 2504 |

# Books

**Edmund Burke, Eric Foxley, Logic and its Applications, John Wiley, 1990, ISBN 0 471 92993**

## Logic

Jon Barwise, John Etchemendy, The Language of First Order Logic, CSLI Lecture Notes, 1993, 0-937073-99-7.

Antony Galton: Logic for Information Technology, John Wiley, 1990. ISBN 0 471 92993 6. (in Library)

## Prolog

I. Bratko, Prolog Programming for Artificial Intelligence, Addison Wesley, 1990, ISBN 0-201-41606-9

W. Clocksin and C. Mellish, Programming in Prolog.

L. Sterling and E. Shapiro, The Art of Prolog, 2nd Edition, MIT Press, 1994.

# What is Logic?

1. Logic is an **analytical tool** (for studying valid forms of argument). Other analytical tools include

   - Ptolemy's Spheres
   - Newtonian Mechanics
   - Catastrophy theory

   *Analytical tools provide concepts, laws and standards within determined areas of enquiry.*

2. A particular logic is a **formal system** within which we can model deductive reasoning. Other formal systems include

   - Unary Arithmetic
   - Chess

   *The sentences of formal systems can be represented and manipulated mechanistically i.e. without regard to meaning.*

3. Logic as a **programming language**. Programming languages include

   - Prolog
   - Pascal
   - Java

   *Programming languages are used to describe computations.*

These three perspectives are intimately connected.

## Logic as a Formal Deductive System

- **Formal Deductive Systems**
  - **Language:** A set of "sentences": i.e. marks on paper.
  - **Rules:** How to generate new sentences from old ones.

- An essential feature of a formal system is that we make no reference to the meanings of the sentences.

- **Example: Unary Numbers**
  - *Language:* The single sentence $\{\mathrm{I}\}$
  - *Rule of concatenation:*

    Given X is a sentence and Y is a sentence, infer that the concatenatation of X and Y is a sentence.

## Why Study Logic within IT?

*Application areas include:*

- Circuit Design
- Semantics of Programming Languages
- Program Specification/Verification
- Knowledge Representation
- Logic Programming

# What is an Argument?

**Definition:** an argument or inference consists of some *premises* and a *conclusion*.

$$\frac{\begin{array}{c}\text{If I'm outside and it's raining then I get wet}\\ \text{I'm outside}\\ \text{It's raining}\end{array}}{\text{Therefore, I get wet}}$$

## Examples of Arguments

(1)

| |
|---|
| Hamish is red in the face |
| Hamish is holding an empty bottle |
| Hamish is shouting |
| Hamish just fell over |
| Therefore: Hamish is drunk |

(2)

| |
|---|
| SLT is in the University |
| The University is in Malta |
| Therefore, SLT is in Malta |

(3)

| |
|---|
| Valletta is not in Italy |
| Italy is not in Australia |
| Therefore, Valletta is not in Australia |

(4)

| |
|---|
| If Valletta is in Malta, I can speak Maltese |
| Valletta is in Malta |
| Therefore, I can speak Maltese |

# Good and Bad Arguments

(1) is a bad argument because the conclusion might be wrong even if the premises are true. For instance, Hamish might be an actor in a play.

(2) is a good argument because if you accept the premises you must also accept the conclusion. Here the conclusion is true.

(3) is a bad argument because even if you accept the premises, the conclusion does not follow. Here the conclusion is true.

(4) is a good argument, but the conclusion is false (in 1998).

The quality or **validity** of an argument does not depend on the particular truth status of either the premises or the conclusion but on the relationship between the two.

# Valid Argument

A *valid* argument is one in which the premises *entail* the conclusion.

An *invalid* argument is one in which they do not.

The validity of an argument is independent of the truth of its premises or its conclusion,

BUT

*If the premises are true, and the argument is valid, then the conclusion must also be true.*

Given a valid argument and true premises, you have the premises, you can *infer* the conclusion.

## What are Premises and Conclusions Made Of?

Answer: Propositions, i.e. idealised abstraction that we use to investigate the logical properties of English sentences.

A *proposition* corresponds to the contents of a sentence

"My name is Mike" or
"Max saw Claire"

We sometimes need to distinguish between proposition and the sentence used to express it.

Mi chiamo Mike
Je m'appelle Mike
Ich heisse Mike
Jisimni Mike

**Essential property of a proposition** is that it is either *true* or *false*.

# Atomic Sentences

Atomic sentences correspond to the simplest declarative sentences of English, and often consist of **names** and **predicates**.

*Claire gave Scruffy to Max*

In first order logic (FOL), atomic sentences can also be formed by combining names (or individual constants) and predicates, albeit according to slightly different rules.

gave(Claire, Scruffy, Max)

Different versions of FOL have different conventions for forming names and predicates and different conventions for for combining them.

## Individual Constants

Individual constants are simply symbols that are used to refer to some fixed individual object. They are the FOL analogue of names.

Mike
1

Names in FOL are quite similar to names in English except that names in FOL must refer to **exactly one** object.

The properties of names in FOL can be summarised as follows:

- Every name must name an object
- No name can name more than one object
- An object can have more than one name, or no name at all.

# Predicate Symbols

Predicate symbols are symbols used to denote some property of objects or some relation between objects. Consider the sentence

Max likes Claire

In FOL, we view this as a claim that the predicate *likes* expresses a relation between the individuals referred to by the names *Max* and *Claire*.

These individuals are called the "arguments". In this case, there are two.

A unary predicate with one argument denotes a property.

A predicate of arity $>1$ denotes a relation.

# Predicates

In English, some predicates have optional arguments.

Claire gave
Claire gave Scruffy
Claire gave Scruffy to Mike

In FOL, the number of arguments a given predicate can take is fixed. Its called the "arity" of the predicate.

In English, predicates are sometimes vague.

FOL assumes that to every predicate their corresponds a determinate property or relation, i.e. one for which given any object or set of objects, there is a definite fact of the matter whether or not the objects have that property or are in that relation.

## Equality and Infix Notation

So far all the atomic sentences we have seen have been formed by a predicate followed by an appropriate number of arguments, e.g.

Taller(Claire, Max)
Taller(Max,Claire)

bf N.B. the order of the arguments matters (most of the time).

Equality is also a binary relation. However, in this case, we usually put the predicate name between the arguments.

$a = b$

## Summary

- Every predicate symbol comes with a single, fixed, arity.
- Every predicate is interpreted by a determinate or relation of the same arity as the predicate.
- Atomic sentences are formed by putting a predicate of arity n in front of n names, or (in the case of infix notation) by putting a binary predicate between a pair of names.
- The order of names is crucial in forming a binary predicate