Discrete Mathematics
CSE 2131

Logic Module

Logic and It's Applications

- Logic provides a powerful tool for reasoning correctly about mathematics, algorithms, and computers.

- It is used extensively throughout computer science, and you need to understand its basic concepts in order to study many of the more advanced subjects in computing.

Logic and It's Applications

- **In software engineering,** it is good practice to specify what a system should do before starting to code it. Logic is frequently used for software specifications.
- **In safety-critical applications,** it is essential to establish that a program is correct. Conventional debugging isn't enough—what we want is a proof of correctness. Formal logic is the foundation of program correctness proofs.
- **In information retrieval, including Web search engines,** logical propositions are used to specify the properties that should (or should not) be present in a piece of information in order for it to be considered relevant.
- **In artificial intelligence,** formal logic is sometimes used to simulate intelligent thought processes. People don't do their ordinary reasoning using mathematical logic, but logic is a convenient tool for implementing certain forms of reasoning.

Logic and It's Applications

- **In digital circuit design and computer architecture,** logic is the language used to describe the signal values that are produced by components. A common problem is that a first-draft circuit design written by an engineer is too slow, so it has to be transformed into an equivalent circuit that is more efficient. This process is often quite tricky, and logic provides the framework for doing it.
- **In database systems,** complex queries are built up from simpler components. It's essential to have a clear, precise way to express such queries, so that users of the database can be sure they're getting the right information. Logic is the key to expressing such queries.
- **In compiler construction,** the type checking phase must determine whether the program being translated uses any variables or functions inconsistently. It turns out that the method for doing this is similar to the method for performing logical inference. As a result, algorithms that were designed originally to perform calculations in mathematical logic are now embedded in modern compilers.

Logic and It's Applications

- **In programming language design,** one of the most commonly used methods for specifying the meaning of a computer program is the lambda calculus, which is actually a formal logical system originally invented by a mathematician for purely theoretical research.

- **In computability theory,** logic is used both to specify abstract machine models and to reason about their capabilities. There has been an extremely close interaction between mathematical logicians and theoretical computer scientists in developing a variety of machine models.

Logic in general

Logics are formal languages for formalizing reasoning, in particular for representing information such that conclusions can be drawn

A logic involves:

- A language with a syntax for specifying what is a legal expression in the language;
- syntax defines well formed sentences in the language
- Semantics for associating elements of the language with elements of some subject matter.
  - Semantics defines the "meaning" of sentences (link to the world); i.e., semantics defines the truth of a sentence with respect to each possible world
- Inference rules for manipulating sentences in the language

Original motivation: Early Greeks settled arguments based on purely rigorous (symbolic/syntactic) reasoning starting from a given set of premises.

Example of a formal language: Arithmetic

E.g., the language of arithmetic

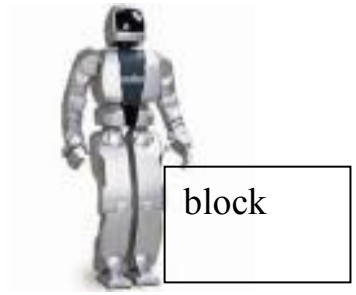
- $x+2 \geq y$ is a sentence;
- $2x+y > \{\}$ is not a sentence
- $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
- $x+2 \geq y$ is true in a world where $x = 7, y = 1$
- $x+2 \geq y$ is false in a world where $x = 0, y = 6$

Simple Robot Domain

Consider a robot that is able to lift a block,

- if that block is liftable (i.e., not too heavy), and
- if the robot's battery power is adequate.

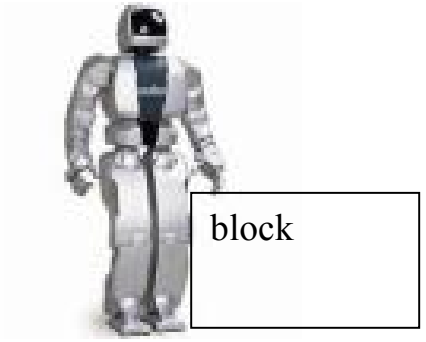
If both of these conditions are satisfied, then when the robot tries to lift a block it is holding, its arm



Feature 1: BatIsOk (True or False)
Feature 2: BlockLiftable (True or False)
Feature 3: RobotMoves (True or False)

Simple Robot Domain

We need a *language* to express
the *features/properties/assertions* and *constraints among them*; also *inference mechanisms*, *i.e*, principled ways of performing reasoning.



Example - logical statement about the robot:

(BatIsOk and BlockLiftable) implies RobotMoves

Binary valued featured descriptions

Consider the following description:

- The router can send packets to the edge system only if it supports the new address space. For the router to support the new address space it is necessary that the latest software release be installed. The router can send packets to the edge system if the latest software release is installed. The router does not support the new address space.

- Features:
  - Router
    - Feature 1 – router can send packets to the edge of system
    - Feature 2 – router supports the new address space
  - Latest software release
    - Feature 3 – latest software release is installed

Binary valued featured descriptions

- Constraints:
  - The router can send packets to the edge system only if it supports the new address space. (constraint between feature 1 and feature 2);
  - It is necessary that the latest software release be installed for the router to support the new address space . (constraint between feature 2 and feature 3);
  - The router can send packets to the edge system if the latest software release is installed. (constraint between feature 1 and feature 3);

How can we write these specifications in a formal language and reason about the system?

1.1 Propositional Logic

Syntax:
Elements of the language

Primitive propositions --- statements like*:*

Bob loves Alice ⟶ P
Alice loves Bob ⟶ Q

Propositional Symbols
(atomic propositions)

Compound propositions

Bob loves Alice *and* Alice loves Bob

⟶ $P \wedge Q$ ($\wedge$ - stands for and)

Connectives

- $\neg$ - not

$\forall$ $\wedge$ - and

$\forall$ $\vee$ - or

$\forall$ $\rightarrow$ - implies

$\forall$ $\leftrightarrow$ - equivalent (if and only if)