Logic as a Programming Language

- Logic can be considered the oldest programming language
- Aristotle invented propositional logic over 2000 years ago in order to prove properties of formal arguments
- Propositions - simple statements that are either true or false; e.g. Betty wears a white dress. Today is Sunday.
- Propositional Logic ≡ propositions + rules of inference
- Most famous inference rule: modus ponens

Let A and B be propositions, then

A implies B
A is true

∴ B is true

HW:
(1) Read Section 1 online tutorial
(2) Install SWI Prolog
Both are available on the CSC301 Prolog page.

(1) **Inference** is the act or process of drawing a conclusion based solely on what one already knows.
(2) **Rule of inference** is a scheme for constructing valid inferences.

Propositional Logic

Example:

If Betty wears a white dress then today is Sunday.
Betty wears a white dress.

∴ Today is Sunday.

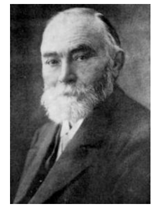
A fundamental problem with propositional logic is that it is not powerful enough to encode general knowledge - we would like to say things like:

All objects that are considered human are mortal.

Due to the fact that this sentence is not simple it can not be considered a proposition. But these kind of sentences are key in describing general knowledge.

Quantification

- In 1879 Gottlob Frege introduced the *predicate calculus* (‘Begriffsschrifft’)
- Today predicate calculus is more commonly known as First Order Logic.
- This logic solves the problems of propositional logic by introducing three new structures: predicates, universal quantification, and existential quantification.



Friedrich Ludwig Gottlob Frege
Philosopher and Logician

First order Logic

(1) Predicates - functions that map their arguments to true or false
e.g. brothers(peter,john).

(2) Universal Quantifier: ∀ (‘for **A**ll objects…’)
e.g. ∀x[P(x)] - for all objects x such that the predicate P(x) is true.

(3) Existential Quantifier: ∃ (‘there **E**xists and object…’)
e.g. ∃x[P(x)] - there exists an object x such that the predicate P(x) is true.

Note: the inference rules such as the modus ponens still hold, consider:

∀x[human(x) implies mortal(x)] human(socrates) ---------------------------- ∴ mortal(socrates)	for all objects x such that if human(x) is true then mortal(x) is true. human(socrates) is true. ---------------------------- ∴ mortal(socrates) is true
	Long text of logic statement to the left

Logical Connectives & WFFs

We can further extend our logic with logical connectives:

- ∧ (and)
- ∨ (or)
- ¬ (not)
- → (implies)

This allows us to write WFF's (Well Formed Formulas)

$\forall x,y[female(x) \wedge parent(x,y) \rightarrow mother(x,y) \wedge child(y,x)]$
$\forall x,y[(female(x) \vee male(x)) \wedge parent(x,y) \rightarrow mother(x,y) \vee father(x,y)]$

It should be clear that general WFF's can become very complex.

Horn Clause Logic

In horn clause logic the form of the WFF's is restricted:

$$P_1 \wedge P_2 \wedge \ldots \wedge P_{n-1} \wedge P_n \rightarrow P_0$$

Single predicate in consequent

Conjunctions only!

Where P_0 , P_1 , P_2, … P_{n-1}, P_n are predicates.

Proving things is computation!

Use resolution to reason with horn clause expressions - resolution mimics the modus ponens using horn clause expressions.

Advantage: this can be done mechanically (Alan Robinson, 1965)

"Deduction is Computation"

J. Alan Robinson: A Machine-Oriented Logic Based on the Resolution Principle. J. ACM 12(1): 23-41 (1965)

Basic Prolog Programs

Facts - a fact constitutes a declaration of a truth; in Prolog it has to to be a positive assertion.

Prolog Programs - a Prolog program is a collection of facts (…and rules, as we will see later).

Example: a simple program

```
male(phil).
male(john).
female(betty).
```

} Facts, Prolog will treat these as true and enter them into its knowledgebase.

We execute Prolog programs by posing queries on its knowledgebase:

Prompt → ?-

```
?- male(phil).
   true - because Prolog can use its knowledgebase to prove true.
?- female(phil).
   false - this fact is not in the knowledgebase.
```

Prolog = Programming in Logic

First-Order Logic

- Quantified Variables
  - <u>Universally</u> quantified variables

    ∀X – <u>for all</u> objects X

  - <u>Existentially</u> quantified variables

    ∃Y – <u>there exists</u> an object Y

First-Order Logic

- Predicates
  - Predicates are functions that map their arguments into true/false
  - The signature of a predicate p(X) is

    p: Objects → { true, false }

  - Example: human(X)
    - human: Objects → { true, false }
    - human(tree) = false
    - human(paul) = true
  - Example: mother(X,Y)
    - mother: Objects × Objects → { true, false }
    - mother(betty,paul) = true
    - Mother(giraffe,peter) = false

First-Order Logic

- We can combine predicates and quantified variables to make statements on sets of objects
  - ∃X[mother(X,paul)]
    - there exists an object X such that X is the mother of Paul
  - ∀Y[human(Y)]
    - for all objects Y such that Y is human

First-Order Logic

- Logical Connectives: and, or, not
  - ∃F∀C[parent(F,C) and male(F)]
    - There exists an object F for all object C such that F is a parent of C and F is male.
  - ∀X[day(X) and (comfortable(X) or not comfortable(X))]
    - For all objects X such that X is a day and X is either comfortable or not.

First-Order Logic

- If-then rules: A → B
  - ∀X∀Y[parent(X,Y) and female(X) → mother(X)]
    - For all objects X and for all objects Y such that if X is a parent of Y and X is female then X is a mother.
  - ∀Q[human(Q) → mortal(Q)]
    - For all objects Q such that if Q is human then Q is mortal.