

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 \equiv propositions + rules of inference
- Most famous inference rule: modus ponens



Let A and B be propositions, then

A implies B

A is true

\therefore B is true

Reading:
MPL chap 19

- (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.

Reading

- MPL Chapter 19

Propositional Logic

Example:

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

\therefore Today is Sunday.

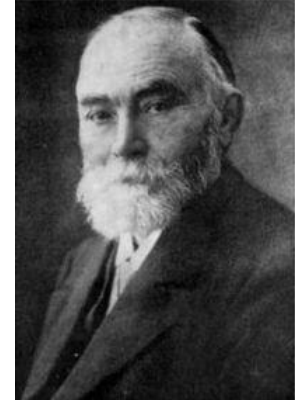
A fundamental problem with propositional logic is that it is not powerful enough to encode general knowledge - we would like to say things like:

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

- o In 1879 Gottlob Frege introduced the *predicate calculus* ('Begriffsschrift')
- o Today predicate calculus is more commonly known as First Order Logic.
- o 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

- Quantified Variables

- Universally quantified variables

$\forall X$ – for All objects X

- Existentially quantified variables

$\exists Y$ – there Exists 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

$p: \text{Objects} \rightarrow \{ \text{true}, \text{false} \}$

- Example: $\text{human}(X)$

- $\text{human}: \text{Objects} \rightarrow \{ \text{true}, \text{false} \}$
- $\text{human}(\text{tree}) = \text{false}$
- $\text{human}(\text{paul}) = \text{true}$

- Example: $\text{mother}(X,Y)$

- $\text{mother}: \text{Objects} \times \text{Objects} \rightarrow \{ \text{true}, \text{false} \}$
- $\text{mother}(\text{betty}, \text{paul}) = \text{true}$
- $\text{mother}(\text{giraffe}, \text{peter}) = \text{false}$

First-Order Logic

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

First-Order Logic

- Logical Connectives: and, or, not
 - $\exists F \forall C [\text{parent}(F, C) \text{ and } \text{male}(F)]$
 - There exists an object F for all object C such that F is a parent of C and F is male.
 - $\forall X [\text{day}(X) \text{ and } (\text{sunny}(X) \text{ or } \text{rainy}(X))]$
 - For all objects X such that X is a day and X is either wet or dry.

First-Order Logic

- If-then rules: $A \rightarrow B$
 - $\forall X \forall Y [\text{parent}(X, Y) \text{ and } \text{woman}(X) \rightarrow \text{mother}(X, Y)]$
 - For all objects X and for all objects Y such that if X is a parent of Y and X is a woman then X is a mother.
 - $\forall Q [\text{human}(Q) \rightarrow \text{mortal}(Q)]$
 - For all objects Q such that if Q is human then Q is mortal.

Logic Formulas

- First-order logic allows you to construct extremely complex statements, e.g.,

$$\forall x \forall y (P(f(x)) \rightarrow \neg(P(x) \rightarrow Q(f(y), x, z)))$$

- These statements are difficult to mechanize.

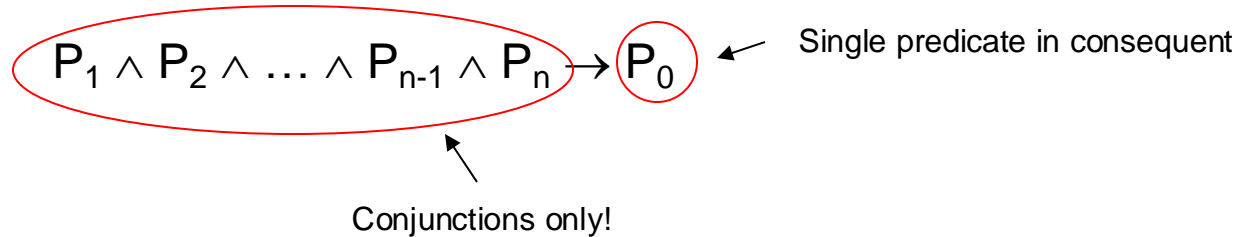
Horn Clause Logic

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

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

Single predicate in consequent

Conjunctions only!

The diagram shows the formula $P_1 \wedge P_2 \wedge \dots \wedge P_{n-1} \wedge P_n \rightarrow P_0$. A red oval encircles the antecedent part $P_1 \wedge P_2 \wedge \dots \wedge P_{n-1} \wedge P_n$, with an arrow pointing to it from the text "Conjunctions only!". Another red circle encircles the consequent part P_0 , with an arrow pointing to it from the text "Single predicate in consequent".

Where $P_0, P_1, P_2, \dots, P_{n-1}, P_n$ 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”

Basic Prolog Programs

Facts - a fact constitutes a declaration of a truth; in Prolog it has 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

```
man(phil).  
man(john).  
woman(betty).
```

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

We execute Prolog programs by posing queries on its knowledgebase:

Prompt → ?- man(phil).
 true - because Prolog can use its knowledgebase to prove true.
 ?- woman(phil).
 false - this fact is not in the knowledgebase.

Prolog - Queries & Goals

A query is a way to extract information from a logic program.

Given a query, Prolog attempts to show that the query is a logical consequence of the program; of the collection of facts.

When queries contain variables, they are existentially quantified, consider

?- parent(X,liz).

The interpretation of this query is: prove that there is at least one object X that can be considered a parent of liz, or formally, prove that

$\exists X[\text{parent}(X,\text{liz})]$

holds.

NOTE: Prolog will return all objects for which a query evaluates to true.

NOTE: A variable in Prolog is a term that starts with a capital letter. Everything else in Prolog has to be written in lower case.

A Prolog Program

```
% a simple prolog program
woman(pam).
woman(liz).
woman(ann).
woman(pat).

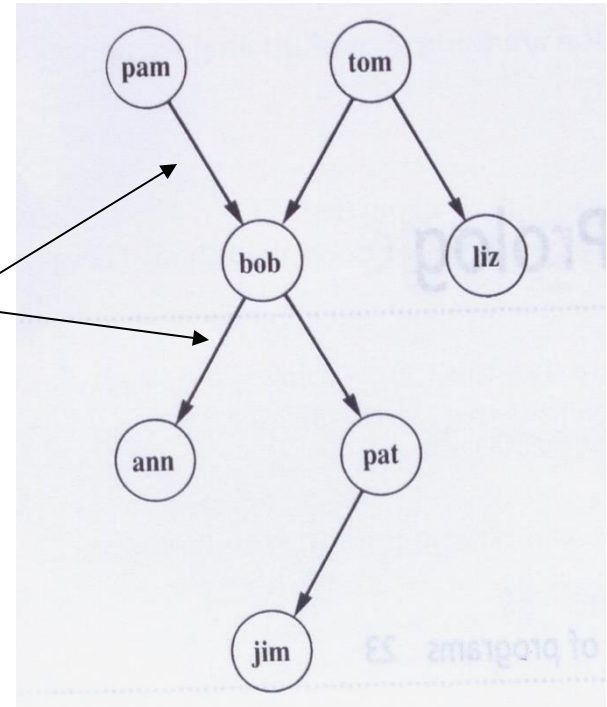
man(tom).
man(bob).
man(jim).

parent(pam,bob).
parent(tom,bob).
parent(tom,liz).
parent(bob,ann).
parent(bob,pat).
parent(pat,jim).
```

Example Queries:

```
?- woman(pam) .
?- woman(X) .            $\exists X [ \text{woman}(X) ] ?$ 
?- parent(tom,Z) .
?- father(Y) .
```

Parent
Relation



A Family Tree

<https://replit.com/@lutzhamel/prolog-csc301#Examples/family.pl>

Demo of 'trace' that demonstrates the search in Prolog

Compound Queries

A compound query is the conjunction of individual simple queries.

Stated in terms of goals: a compound goal is the conjunction of individual subgoals each of which needs to be satisfied in order for the compound goal to be satisfied. Consider:

?- parent(X,Y) , parent(Y,ann).

or formally,

$\exists X,Y[\text{parent}(X,Y) \wedge \text{parent}(Y,\text{ann})]$

When Prolog tries to satisfy this compound goal, it will make sure that the two Y variables always have the same values.

Prolog uses unification and backtracking in order to find all the solutions which satisfy the compound goal.

Unification & Backtracking

- Unification is a special kind of pattern matching that instantiates variables with terms/objects.
- Backtracking allows Prolog to search for all unifications, called substitutions, that make a query true.

Reading

- MPL Chapter 19