Logic Programming

- Declarative programming style
- Example: Prolog
  - Fundamental concept: First-order Predicate Logic
  - The execution of a Prolog program is a proof
  - A proof is an execution of a Prolog program

Horn Clauses

- Conjunctive Normal Form
  - Conjunction of clauses, clauses are disjunction of literals
  - \((A \lor B) \land (C \lor D \lor E)\)...

- Horn Clauses
  - At most one literal is positive, the others are negative

- Definite Horn Clauses
  - Exactly one positive literal
  - \(A \lor \neg B \lor \neg C \lor \neg D\)...
  - \(A \leftarrow B \land C \land D\)...

Facts

- Facts model a relation between elements.
- Facts are definite Horn clauses without negative literals

- Example:
  - olympics.
  - olympics(1896, athens).
  - olympics(1908, london).
  - olympics(2012, london).
  - olympics(2020, tokyo).
Rules

Conditional expressions of the form

\[ A :\neg B, C, D, \ldots Z \]

Semantics: A becomes true if B is true and C is true, ...

Example:

\[ \text{hostCity}(X) :\neg \text{olympics}(_, X). \]

\[ \text{megaHostCity}(\text{City}) :\neg \text{olympics}(_, \text{City}), \]

\[ \text{population}(	ext{City}, \text{Size}), \text{Size} > 5000000. \]

Variables need to start with capital letter or underscore.

'_' means "don't care"

Queries

Queries are expressions that Prolog should try to proof by binding the free variable in such a way that the expression becomes true.

Queries are negative Horn clauses

\[ \leftarrow A \land B \land \ldots \]

Example:

\[ \text{olympics}(1906, \text{london}). \]

\[ \text{olympics}(\text{X}, \text{atlanta}). \]

\[ \text{olympics}(1896, \text{Y}). \]

\[ \text{olympics}(\text{Z}, \text{athens}). \]

Notation

\[ <\text{predicate\_name}>/<\text{arity}> \]

\[ \text{olympics}/0, \text{olympics}/2 \]

Predicate Description

\[ + \]

Argument must be fully instantiated to a term that satisfies the required argument type. Think of the argument as input.

\[ - \]

Argument must be unbound. Think of the argument as output.

\[ ? \]

Argument must be bound to a partial term of the indicated type. Note that a variable is a partial term for any type. Think of the argument as either input or output or both input and output

From the SWI Prolog Manual
Resolution
- gradStudent(bill).
- gradStudent(sally).
- newborn(tom).
- newborn(mary).
- father(bill, mary).
- father(bill, joe).
- mother(sally, tom).
- parent(X, Y) :- father(X, Y).
- parent(X, Y) :- mother(X, Y).
- tired(X) :- gradStudent(X), parent(X, Y), newborn(Y).

SLD-Resolution
- tired(Z)
  - tired(X) :- gradStudent(X), parent(X, Y), newborn(Y).
  - gradStudent(Z), parent(Z, Y), newborn(Y).
    - gradStudent(bill).
    - parent(bill, Y), newborn(Y).
      - parent(X, Y) :- father(X, Y).
      - father(bill, Y), newborn(Y).
        - father(bill, sally).
        - newborn(joe).
        - newborn(mary).
    - gradStudent(sally).
    - parent(sally, Y), newborn(Y).
      - parent(X, Y) :- father(X, Y).

mgu: \( \Theta_1 = \{X \rightarrow Z\} \)
mgu: \( \Theta_2 = \{Z \rightarrow bill\} \)
mgu: \( \Theta_3 = \{X \rightarrow bill\} \)
mgu: \( \Theta_4 = \{Y \rightarrow mary\} \)
Lists

- The empty list is a list.
- A list can be described as [Head | Tail] where Head is an element and Tail is a list.

- We write:
  - []
  - [a,b,c]
    - this is the same as [a] [b] [c]

- List membership:
  - member(X, [X|T]).
  - member(X, [Y|T]) :- member(X, T).

Appending lists:

- append([], X, X).
- append([X|L1], Y, [X|L2]) :- append(L1, Y, L2).

Failure-driven Loops

- olympics(X, athens), write('athens: '), writeln(X), fail; true.

Conditions

- olympics(X,Y), writeln(X), Y==atlanta -> writeln(Y).
- olympics(X,Y), Y == athens -> writeln(' in greece'); writeln(' somewhere else').
The Cut operator

- `olympics(X, Y), write(Y), write(' : '), writeln(X), Y==atlanta -> !, fail; true.`