**Prolog**

- Prolog: "Programming in Logic" (PROgrammation en LOgique)
- One (and maybe the only one) successful logic programming languages
- Useful in AI applications, expert systems, natural language processing, database query languages
- Declarative instead of procedural: "What" instead of "How"

**History of Prolog**

- First Prolog interpreter by Colmerauer and Roussel
- Implementation of DEC10 compiler by Warren
- Definite Clause Grammars implementation by Pereira and Warren

**SWI-Prolog**

- Available for: Linux, Windows, MacOS
History of Prolog

Prolog grows in popularity especially in Europe and Japan


Prolog used to program natural language interface in International Space Station by NASA

Logic Programming

• Program
  Axioms (facts): true statements
• Input to Program
  query (goal): statement true (theorems) or false?
• Thus
  Logic programming systems = deductive databases
datalog

Example

• Axioms:
  0 is a natural number. (Facts)
  For all x, if x is a natural number, then so is the successor of x.
• Query (goal).
  Is 2 a natural number? (can be proved by facts)
  Is -1 a natural number? (cannot be proved)

Another example

• Axioms:
  The factorial of 0 is 1. (Facts)
  If m is the factorial of n - 1, then n * m is the factorial of n.
• Query:
  The factorial of 2 is 3?

First-Order Predicate Calculus

• Logic used in logic programming:
  First-order predicate calculus
  First-order predicate logic
  Predicate logic
  First-order logic
  ∀x (x ≠ x+1)
• Second-order logic
  ∀S ∀x (x ∈ S ∨ x ∈ S)

Lecture 21 – Prolog, Spring 2008
CSE3102 Programming Languages, UT Arlington
©Chengkai Li, 2008
First-Order Predicate Calculus: Example

- natural(0).
  ∀ X, natural(X) → natural(successor(X)).

- ∀ X and Y, parent(X,Y) → ancestor(X,Y).
  ∀ A, B, and C, ancestor(A,B) and ancestor(B,C) → ancestor(A,C).
  ∀ X and Y, mother(X,Y) → parent(X,Y).
  father(bill, jill).
  father(bob, sam).

- factorial(0,1).
  ∀ N and M, factorial(N-1,M) → factorial(N,N*M).

First-Order Predicate Calculus: statements

- Quantifiers
  universal quantifier “for all” ∀
  existential quantifier “there exists” ∃
  bound variable (a variable introduced by a quantifier)
  free variable

- Punctuation symbols
  parentheses (for changing associativity and precedence.)
  comma
  period

- Arguments to predicates and functions can only be terms:
  - Contains constants, variables, and functions.
  - Cannot have predicates, quantifiers, or connectives.

First-Order Predicate Calculus: statements (cont’d)

- The holy grail: we specify the logic itself, the system proves:
  - Not totally realized by logic programming languages. Programmers must be aware of how the system proves, in order to write efficient, or even correct programs.
  - Prove goals from facts:
    - Resolution and Unification

Problem Solving

- Program = Data + Algorithms
- Program = Object.Message( Object)
- Program = Functions Functions
- Algorithm = Logic + Control

Horn Clause

- First-order logic too complicated for an effective logic programming system.
- Horn Clause: a fragment of first-order logic
  b ← a1 and a2 and a3 ... and an.

  Variables in head: universally quantified
  Variables in body only: existentially quantified
  Need “or” in head? Multiple clauses

Horn Clauses: Example

- First-Order Logic:
  natural(0).
  ∀ X, natural(X) → natural(successor(X)).

- Horn Clause:
  natural(0).
  natural(successor(x)) ← natural(X).
Horn Clauses: Example

- First-Order Logic:
  \[
  \text{factorial}(0,1). \\
  \forall N \land \forall M, \text{factorial}(N-1,M) \rightarrow \text{factorial}(N,N*M).
  \]

- Horn Clause:
  \[
  \text{factorial}(0,1). \\
  \text{factorial}(N,N*M) \leftarrow \text{factorial}(N-1,M).
  \]

Horn Clauses: Example

- Horn Clause:
  \[
  \text{ancestor}(X,Y) \leftarrow \text{parent}(X,Y). \\
  \text{ancestor}(A,C) \leftarrow \text{ancestor}(A,B) \land \text{ancestor}(B,C). \\
  \text{parent}(X,Y) \leftarrow \text{mother}(X,Y). \\
  \text{parent}(X,Y) \leftarrow \text{father}(X,Y). \\
  \text{father}(\text{bill},\text{jill}). \\
  \text{mother}(\text{jill},\text{sam}). \\
  \text{father}(\text{bob},\text{sam}).
  \]

Prolog syntax

- \( \text{:-} \) for \( \leftarrow \):
  \[
  \text{ancestor}(X,Y) \leftarrow \text{parent}(X,Y). \\
  \text{ancestor}(X,Y) \leftarrow \text{ancestor}(X,Z) \land \text{ancestor}(Z,Y). \\
  \text{parent}(X,Y) \leftarrow \text{mother}(X,Y). \\
  \text{parent}(X,Y) \leftarrow \text{father}(X,Y). \\
  \text{father}(\text{bill},\text{jill}). \\
  \text{mother}(\text{jill},\text{sam}). \\
  \text{father}(\text{bob},\text{sam}).
  \]

Resolution and Unification

Resolution

- Resolution: Using a clause, replace its head in the second clause by its body, if they "match".
  \[
  a \leftarrow a_1, \ldots, a_n. \\
  b \leftarrow b_1, \ldots, b_m. \\
  \text{if } b \text{ matches } a; \\
  b \leftarrow b_1, \ldots, a_1, \ldots, a_n, \ldots, b_m.
  \]
Resolution: Another view

• Resolution: Combine two clauses, and cancel matching statements on both sides.

\[ a \leftarrow a_1, \ldots, a_n, \]
\[ b \leftarrow b_1, \ldots, b_m, \]
\[ \not\exists \quad b \leftarrow a_1, \ldots, a_n, b_1, \ldots, b_m. \]

Problem solving in logic programming systems

• Program:
  - Statements/Facts (clauses).
  - Goals:
    - Headless clauses, with a list of subgoals.

• Problem solving by resolution:
  - Matching subgoals with the heads in the facts, and replacing the subgoals by the corresponding bodies.
  - Cancelling matching statements.
  - Recursively do this, till we eliminate all goals. (Thus original goals proved.)

Example

• Program:
  - \text{mammal} (\text{human}).
• Goal:
  \[ \leftarrow \text{mammal} (\text{human}). \]
• Proving:
  \[ \text{mammal} (\text{human}) \leftarrow \text{mammal} (\text{human}). \]

Example

• Program:
  - \text{legs}(X,2) \leftarrow \text{mammal}(X), \text{arms}(X,2).
  - \text{legs}(X,4) \leftarrow \text{mammal}(X), \text{arms}(X,0).
  - \text{mammal}(\text{horse}).
  - \text{arms}(\text{horse}, 0).
• Goal:
  \[ \leftarrow \text{legs}(\text{horse}, 4). \]
• Proving:
  \[ ? \]

Unification

• Unification: Pattern matching to make statements identical (when there are variables).
• Set variables equal to patterns: instantiated.
• In previous example:
  - \text{legs}(X,4) and \text{legs}(\text{horse},4) are unified.
  - (X is instantiated with \text{horse}.)

Unification: Example

• Euclid’s algorithm for greatest common divisor
• Program:
  \[ \text{gcd}(V, U, W) \leftarrow \text{not zero}(V), \text{gcd}(V, U \mod V, W). \]
• Goals:
  \[ \leftarrow \text{gcd}(15, 10, X). \]
Things unspecified

• The order to resolve subgoals.
• The order to use clauses to resolve subgoals.
• Possible to implement systems that don’t depend on the order, but too inefficient.
• Thus programmers must know the orders used by the language implementations. (Search Strategies)

Example

• Program:
  ancestor(X,Y) :- ancestor(X,Z), parent(Z,Y).
  ancestor(X,Y) :- parent(X,Y).
  parent(X,Y) :- mother(X,Y).
  parent(X,Y) :- father(X,Y).
  father(bill,jill).
  mother(jill,sam).
  father(bob,sam).

• Goals:
  ← ancestor(bill,sam).
  ← ancestor(X,bob).