Logic Programming (PLP 11.3)
Horn Clauses
Introduction to Prolog: Resolution, Unification

Carlos Varela
Rensselaer Polytechnic Institute

November 9, 2006

An Early (1971) “Conversation”

USER: Cats kill mice.
Tom is a cat who does not like mice who eat cheese.
Jerry is a mouse who eats cheese.
Max is not a mouse.
What does Tom do?
COMPUTER: Tom does not like mice who eat cheese.
Tom kills mice.
USER: What does Jerry eat?
COMPUTER: Cheese.
USER: Who does not like mice who eat cheese?
COMPUTER: Tom.
USER: What does Tom eat?

Logic programming

• A program is a collection of axioms, from which theorems can be proven.
• A goal states the theorem to be proved.
• A logic programming language implementation attempts to satisfy the goal given the axioms and built-in inference mechanism.

Horn clauses

• A standard form for writing axioms, e.g.:

  father(X,Y) ← parent(X,Y), male(X).

• The Horn clause consists of:
  – A head or consequent term H, and
  – A body consisting of terms B_i

  H ← B_0, B_1, ..., B_n

• The semantics is:

  « If B_0, B_1, ..., B_n, then H »

Terms

• Constants
  rpi
  troy

• Variables
  University
  City

• Predicates
  located_at(rpi, troy)
  pair(a, pair(b, c))
Resolution

• To derive new statements, Robinson’s resolution principle says that if two Horn clauses:

\[ H_1 \leftarrow B_{11}, B_{12}, \ldots, B_{1n} \]
\[ H_2 \leftarrow B_{21}, B_{22}, \ldots, B_{2m} \]

are such that \( H_1 \) matches \( B_{2i} \), then we can replace \( B_{2i} \) with \( B_{11}, B_{12}, \ldots, B_{1n} \):

\[ H_2 \leftarrow B_{21}, B_{22}, \ldots, B_{2(i-1)}, B_{11}, B_{12}, \ldots, B_{1n}, B_{2(i+1)} \ldots, B_{2m} \]

• For example:

\[
\begin{align*}
C & \leftarrow A, B \\
D & \leftarrow C \\
\end{align*}
\]

\[
\frac{D \leftarrow A, B}{D \leftarrow A, B}
\]

Resolution Example

\[
\begin{align*}
\text{father}(X,Y) & :- \text{parent}(X,Y), \text{male}(X). \\
\text{ancestor}(X,Y) & :- \text{father}(X,Y). \\
\text{ancestor}(X,Y) & :- \text{parent}(X,Y), \text{male}(X). \\
\end{align*}
\]

Unification

• During resolution, free variables acquire values through unification with expressions in matching terms.

• For example:

\[
\begin{align*}
\text{male(carlos)}. \\
\text{parent(carlos, tatiana)}. \\
\text{father}(X,Y) & :- \text{parent}(X,Y), \text{male}(X). \\
\text{father}(\text{carlos}, \text{tatiana}). \\
\end{align*}
\]

Unification Process

• A constant unifies only with itself.

• Two predicates unify if and only if they have
  – the same functor,
  – the same number of arguments, and
  – the corresponding arguments unify.

• A variable unifies with anything.
  – If the other thing has a value, then the variable is instantiated.
  – If it is an uninstantiated variable, then the two variables are associated.

✓ Just like compatible assignment of single-assignment variables in Oz.

Prolog lists

• \([a,b,c] \) is syntactic sugar for:

\[
.(a., b., (c, [])))
\]

where \( [] \) is the empty list, and \( . \) is a built-in cons-like functor.

• \([a,b,c] \) can also be expressed as:

\[
\begin{align*}
[a | [b, c | []]], \text{or} \\
[a, b | [c | []]], \text{or} \\
[a, b, c | []]
\end{align*}
\]

Prolog lists append example

\[
\begin{align*}
\text{append([],L,L)}. \\
\text{append([H|T],A,[H,L])} & :- \text{append}(T,A,L).
\end{align*}
\]
Backtracking

- **Forward chaining** goes from axioms forward into goals.
- **Backward chaining** starts from goals and works backwards to prove them with existing axioms.

Backtracking example

```
rainy(seattle).
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X).
```

Backtracking example

```
rainy(seattle).
rainy(rochester).
cold(rochester).
snowy(X) :- rainy(X), cold(X).
```

Propositional Logic

- Assigning truth values to logical propositions.
- Formula syntax:
  
  \[
  f : = \upsilon \quad \text{symbol} \\
  | f \land f \quad \text{and} \\
  | f \lor f \quad \text{or} \\
  | f \equiv f \quad \text{if and only if} \\
  | f \implies f \quad \text{implies} \\
  | \neg f \quad \text{not}
  \]
Truth Values

• To assign a truth values to a propositional formula, we have to assign truth values to each of its atoms (symbols).

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a ∨ b</th>
<th>a ⇔ b</th>
<th>¬ a</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

Predicate Calculus

• In mathematical logic, a predicate is a function that maps constants or variables to true and false.

• Predicate calculus enables reasoning about propositions.

For example:

∀C [rainy(C) ∧ cold(C) ⇒ snowy(C)]

Quantifiers

• Universal (∀) quantifier indicates that the proposition is true for all variable values.

• Existential (∃) quantifier indicates that the proposition is true for at least one value of the variable.

• For example:

∀A ∀B [∃C [takes(A, C) ∧ takes(B, C)] ⇒ classmates(A, B)]
Clausal Form

- Looking for a minimal kernel appropriate for theorem proving.
- Propositions are transformed into normal form by using structural congruence relationship.
- One popular normal form candidate is clausal form.
- Clocksin and Melish (1994) introduce a 5-step procedure to convert first-order logic propositions into clausal form.

Clocksin and Melish Procedure

1. Eliminate implication ($\Rightarrow$) and equivalence ($\equiv$).
2. Move negation ($\neg$) inwards to individual terms.
3. Skolemization: eliminate existential quantifiers ($\exists$).
4. Move universal quantifiers ($\forall$) to top-level and make implicit, i.e., all variables are universally quantified.
5. Use distributive, associative and commutative rules of $\lor$, $\land$, and $\neg$, to move into conjuctive normal form, i.e., a conjunction of disjunctions (or clauses).

Example

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \neg \exists B \ [\text{takes}(A,B) \land \text{class}(B)]]$$

1. Eliminate implication ($\Rightarrow$) and equivalence ($\equiv$).

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \neg \exists B \ [\text{takes}(A,B) \land \text{class}(B)]]$$

2. Move negation ($\neg$) inwards to individual terms.

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

3. Skolemization: eliminate existential quantifiers ($\exists$).

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

4. Move universal quantifiers ($\forall$) to top-level and make implicit, i.e., all variables are universally quantified.

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

5. Use distributive, associative and commutative rules of $\lor$, $\land$, and $\neg$, to move into conjuctive normal form, i.e., a conjunction of disjunctions (or clauses).

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

Example Continued

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

6. Use commutativity of $\lor$ to move negated terms to the right of each clause.

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

7. Use $P_1 \lor P_2 \Rightarrow P_1 \lor P_2 \Rightarrow P_2$

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

8. Move Horn clauses to Prolog:

$$\forall A \ [\neg \text{dorm_resident}(A) \lor \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)]]$$

Clausal Form to Prolog

$$\text{student}(A) \ :- \ \text{dorm_resident}(A).$$
$$\text{student}(A) \ :- \ \text{takes}(A,B),\text{class}(B).$$

Skolemization

introduce a Skolem constant to get rid of existential quantifier ($\exists$):

$$\exists A \ [\text{campus_address_of}(X,A)]$$

introduce a Skolem function to get rid of existential quantifier ($\exists$):

$$\forall X \ [\neg \text{dorm_resident}(X) \lor \forall A \ [\text{campus_address_of}(X,A)]]$$
Limitations

- If more than one non-negated (positive) term in a clause, then it cannot be moved to a Horn clause (which restricts clauses to only one head term).
- If zero non-negated (positive) terms, the same problem arises (Prolog’s inability to prove logical negations).
- For example:
  - « every living thing is an animal or a plant »
    
    \[
    \text{animal}(X) \lor \text{plant}(X) \lor \neg \text{living}(X)
    \]
    
    \[
    \text{animal}(X) \lor \text{plant}(X) \iff \text{living}(X)
    \]

Exercises

70. What is the logical meaning of the second Skolemization example if we do not introduce a Skolem function?
71. Download Prolog and execute the "snowy(City)" example. Use "tracing" to follow backtracking step by step.
72. PLP Exercise 11.21 (pg 654).
73. *PLP Exercise 11.23 (pg 654).