Logic Programming (PLP 11)
Predicate Calculus, Horn Clauses,
Clocksin-Mellish Procedure

Carlos Varela
Rensselaer Polytechnic Institute

November 8, 2019
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:  
Who is a cat?  

COMPUTER:  
Tom.

USER:  
What does Jerry eat?  

COMPUTER:  
Cheese.  

USER:  
Who does not like mice who eat cheese?  

COMPUTER:  
Tom.  

USER:  
What does Tom eat?  

COMPUTER:  
What cats who do not like mice who eat cheese eat.

C. Varela
Another Conversation

USER:

Every psychiatrist is a person.
Every person he analyzes is sick.
Jacques is a psychiatrist in Marseille.
Is Jacques a person?
Where is Jacques?
Is Jacques sick?

COMPUTER:

Yes.
In Marseille.
I don’t know.
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.
Propositional Logic

• Assigning truth values to logical propositions.
• Formula syntax:

\[
\begin{align*}
\text{f} & : = \  v & \text{symbol} \\
| & \ f \ & \ f & \text{and} \\
| & \ f \ & \ f & \text{or} \\
| & \ f \ & \ f & \text{if and only if} \\
| & \ f \ & \ f & \text{implies} \\
| & \ f & \ & \text{not}
\end{align*}
\]
Truth Values

- To assign a truth value to a propositional formula, we have to assign truth values to each of its atoms (symbols).
- Formula semantics:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>a \land b</th>
<th>a \lor b</th>
<th>a \iff b</th>
<th>a \Rightarrow b</th>
<th>\neg a</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Tautologies

- A *tautology* is a formula, *true* for all possible assignments.

- For example: \( \neg\neg p \iff p \)

- The contrapositive law:

  \[(p \implies q) \iff (\neg q \implies \neg p)\]

- De Morgan’s law:

  \[\neg (p \land q) \iff (\neg p \lor \neg q)\]
First Order Predicate Calculus

- Adds variables, terms, and (first-order) quantification of variables.
- Predicate syntax:

\[
a ::= p(v_1, v_2, ..., v_n) \quad \text{predicate}
\]
\[
f ::= a \quad \text{atom}
\mid v = p(v_1, v_2, ..., v_n) \quad \text{equality}
\mid v_1 = v_2
\mid f \land f \mid f \lor f \mid f \leftrightarrow f \mid f \Rightarrow f \mid \neg f
\mid \forall v . f \quad \text{universal quantifier}
\mid \exists v . f \quad \text{existential quantifier}
\]
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:

\[ \forall C \ [\text{rainy}(C) \land \text{cold}(C) \Rightarrow \text{snowy}(C)] \]

Quantifier (\( \forall, \exists \))

Operators (\( \land, \lor, \iff, \Rightarrow, \neg \))
Quantifiers

- **Universal** (\(\forall\)) quantifier indicates that the proposition is true for **all** variable values.

- **Existential** (\(\exists\)) quantifier indicates that the proposition is true for **at least one** value of the variable.

- For example:

\[
\forall A \ \forall B \ [ (\exists C \ [ \ \text{takes}(A,C) \ \land \ \text{takes}(B,C) ]) \ \Rightarrow \ \text{classmates}(A,B) ]
\]
Structural Congruence Laws

\[ P_1 \implies P_2 \equiv \neg P_1 \lor P_2 \]

\[ \neg \exists X \ [P(X)] \equiv \forall X \ [\neg P(X)] \]
\[ \neg \forall X \ [P(X)] \equiv \exists X \ [\neg P(X)] \]

\[ \neg (P_1 \land P_2) \equiv \neg P_1 \lor \neg P_2 \]
\[ \neg (P_1 \lor P_2) \equiv \neg P_1 \land \neg P_2 \]
\[ \neg \neg P \equiv P \]

\[ (P_1 \iff P_2) \equiv (P_1 \implies P_2) \land (P_2 \implies P_1) \]

\[ P_1 \lor (P_2 \land P_3) \equiv (P_1 \lor P_2) \land (P_1 \lor P_3) \]
\[ P_1 \land (P_2 \lor P_3) \equiv (P_1 \land P_2) \lor (P_1 \land P_3) \]

\[ P_1 \lor P_2 \equiv P_2 \lor P_1 \]
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 (⇒) and equivalence (⇔).
2. Move negation (¬) inwards to individual terms.
3. **Skolemization**: eliminate existential quantifiers (∃).
4. Move universal quantifiers (∀) to top-level and make implicit, i.e., all variables are universally quantified.
5. Use distributive, associative and commutative rules of ∨, ∧, and ¬, to move into *conjunctive normal form*, i.e., a conjunction of disjunctions (or *clauses*.)
Example

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

1. Eliminate implication (\(\Rightarrow\)) and equivalence (\(\Leftrightarrow\)).

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

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

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

\[ \forall A \ [ \text{student}(A) \lor (\neg \text{dorm_resident}(A) \land \forall B \ [\neg \text{takes}(A,B) \lor \neg \text{class}(B)])] \]
Example Continued

∀A \ [\text{student}(A) \lor (\neg \text{dorm\_resident}(A) \land \forall B [\neg \text{takes}(A,B) \lor \neg \text{class}(B)])]

3. **Skolemization**: eliminate existential quantifiers (∃).

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

   \text{student}(A) \lor (\neg \text{dorm\_resident}(A) \land \neg \text{takes}(A,B) \lor \neg \text{class}(B))

5. Use distributive, associative and commutative rules of ∨, ∧, and ¬, to move into *conjunctive normal form*, i.e., a conjunction of disjunctions (or *clauses*).

   (\text{student}(A) \lor \neg \text{dorm\_resident}(A)) \land
   (\text{student}(A) \lor \neg \text{takes}(A,B) \lor \neg \text{class}(B))
Horn clauses

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

\[ \text{father}(X,Y) \iff \text{parent}(X,Y), \text{male}(X). \]

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

\[ H \iff B_0, B_1, \ldots, B_n \]

• The semantics is:

« If \( B_0, B_1, \ldots, \) and \( B_n \), then \( H \) »
Clausal Form to Prolog

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

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

7. Use \(P_1 \lor \neg P_2 \equiv P_2 \Rightarrow P_1 \equiv P_1 \iff P_2\)

\[(\text{student}(A) \iff \text{dorm_resident}(A)) \land \]
\[(\text{student}(A) \iff \neg (\neg \text{takes}(A,B) \lor \neg \text{class}(B)))\]

8. Move Horn clauses to Prolog:

\[
\begin{align*}
\text{student}(A) & : - \text{dorm_resident}(A). \\
\text{student}(A) & : - \text{takes}(A,B), \text{class}(B).
\end{align*}
\]
Skolemization

$$\exists X \ [\text{takes}(X, \text{cs101}) \land \text{class\_year}(X,2)]$$

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

$$\text{takes}(x, \text{cs101}) \land \text{class\_year}(x,2)$$

$$\forall X \ [\neg \text{dorm\_resident}(X) \lor$$
$$\exists A \ [\text{campus\_address\_of}(X,A)]]$$

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

$$\forall X \ [\neg \text{dorm\_resident}(X) \lor$$
$$\text{campus\_address\_of}(X,f(X))]$$

C. Varela
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

72. What is the logical meaning of the second Skolemization example if we do not introduce a Skolem function?

73. Convert the following predicates into Conjunctive Normal Form, and if possible, into Horn clauses:
   a) \( \forall C \ [\text{rainy}(C) \land \text{cold}(C) \Rightarrow \text{snowy}(C)] \)
   b) \( \exists C \ [\neg \text{snowy}(C)] \)
   c) \( \neg \exists C \ [\text{snowy}(C)] \)

74. PLP Exercise 11.5 (pg 571).

75. PLP Exercise 11.6 (pg 571).