Prolog As A Theorem Prover
Talk in Automated Reasoning Systems

Jakob Praher

2011-06-07
Deductive Reasoning

Figure: General Picture
Formal Deductions

- **Logic Calculus**
  - Set of axioms (A): Formulae assumed to be true
  - Set of formulae (Γ)
  - Rules of inference: Obtain new formulae from given ones

- **Theorems of a Logic Calculus**
  - The set of formulae obtained by rules of inference from Γ ∪ A
  - Formal: \( \{ \phi \mid \Gamma \vdash \phi \} \)
  - Deduction: a set sequence of formulae recording how \( \phi \) was obtained from Γ ∪ A

- **Not unique**
  - Different calculi exist (E.g. distinct sets of axioms and rules of inference)
Deductive Reasoning in Prolog

Figure: Prolog
Example Problem: Reachable Vertices in a Graph

- Situation (facts about the problem domain):

  Figure: Graph with vertices (a,b,c,d,e) and directed edges.

- Problem: Is there a path starting from c?
Abstract Solution in Predicate Logic

Knowledge (Formalized situation)

edge(a,b), edge(a,c), edge(b,d), edge(c,d), edge(d,e),

∀_{S,E} \text{edge}(S, E) \Rightarrow \text{path}(S, E)),

∀_{S,E} \exists_N (\text{edge}(S, N) \land \text{path}(N, E)) \Rightarrow \text{path}(S, E))

Goal (Problem)

∃_X \text{path}(c, X).
Abstract Solution as Prolog Horn Clauses

Clausal Form

Description of situation

\[
\begin{align*}
edge(a, b) & \leftarrow \top \\
edge(a, c) & \leftarrow \top \\
edge(b, d) & \leftarrow \top \\
edge(c, d) & \leftarrow \top \\
edge(d, e) & \leftarrow \top \\
path(S, E) & \leftarrow edge(S, E) \\
path(S, E) & \leftarrow edge(S, N), path(N, E)
\end{align*}
\]

Problem

\[path(c, X)\]
Derivation Tree with Fixed Atom Selection
Limitations of Prolog as general Prover

- Formal Language: Horn Logic
  - Restricted form of first order predicate logic.
  - At most one positive literal
- Negation as failure
  - No distinction between failed derivation and something being false.
- Depth first strategy:
- Clark’s completion:
Classification of Proof Methods

- **Forward-reasoning (local, bottom-up)**
  Start from the assumptions (axioms) until the conjecture is reached.
  - Resolution method (Robinson 1965)
  - Inverse method (Maslov, Nauk 1964)

- **Goal-oriented (global, top-down)**
  Start from the conjecture until we reach the axioms.
  Grows the tree prove tree upward.
  - Linear resolution (SLD, Prolog)
  - Model elimination method (Loveland 1968)
  - Tableau method
Full FOPL Theorem Provers in Prolog

- **Prolog-like (compilation to Lisp):**
  - PTTP: Prolog technology theorem prover: Uses model elimination (Loveland) (forward-reasoning)

- **Lean theorem provers (Running on top of Prolog):**
  - Satchmo: Tableau proof procedure (bottom-up, forward-reasoning)
  - leanTap: Lean semantic tableau theorem prover (bottom-up, forward reasoning)
  - **leanCoP:** Lean Connection-Based Theorem Prover (top-down, goal-oriented)
Connection Method Concepts

Propositional Case, Formula:
\[(U \land V \land \neg W) \lor (U \land W \land \neg X) \lor \neg U \lor X \lor \neg V\]

Matrix

\[
\begin{array}{ccc}
U & U & \\
-U & V & W & X & \neg V \\
-W & \neg X & \\
\end{array}
\]

Path

\[
\begin{array}{cccc}
-U & U & -U & V & W & X & \neg V \\
-U & V & W & X & -V & \\
-W & \neg X & \\
\end{array}
\]

,  

\[
\begin{array}{cccc}
-U & V & W & X & -V \\
-U & V & W & X & -V \\
-W & \neg X & \\
\end{array}
\]

,  

\[
\begin{array}{cccc}
-U & V & W & X & -V \\
-U & V & W & X & -V \\
-W & \neg X & \\
\end{array}
\]

,  

\[
\begin{array}{cccc}
-U & V & W & X & -V \\
-U & V & W & X & -V \\
-W & \neg X & \\
\end{array}
\]

,  

\[
\begin{array}{cccc}
-U & V & W & X & -V \\
-U & V & W & X & -V \\
-W & \neg X & \\
\end{array}
\]

,  

......
Connection Method Concepts (2)

Connection
A connection in a matrix is an unordered pair of occurrences of complementary literals.

Complementary Path
A connection in a matrix is an unordered pair of occurrences of complementary literals.

Spanning Set of Connections
A set of connections in a matrix if every path through the matrix contains at least one of the connections belonging to this set.
Theorem
A formula of propositional logic in disjunctive normal form (DNF) is valid iff every path through its matrix representation contains connections (is complementary).

= A formula of propositional logic in DNF is valid iff the set of all connections in its matrix is spanning.
Connection Method in First Order Logic

- Extension is done to a possible new variant of a clause (variable renaming)

Example: \((a) \land (\forall x p(x) \Rightarrow p(f(x)) \Rightarrow p(f(f(a))))\)

E.g. \([p(a)], [\neg p(f(f(a)))]\), \([\neg p(X), p(f(X))]\)

- Connections must be compatible (MGU of the set of connections)

- Does not terminate on all inputs
Connection Calculus

Let \((C, M, P)\) be \((\text{DNF-clause}, \text{set of clauses in DNF, the path})\).

\[
\text{axiom} \quad \frac{}{(\emptyset, M, P)}
\]

for some positive \(C \in M\):

\[
\text{start rule} \quad \frac{(C, M \setminus C, \emptyset)}{M}
\]

for some \(L \in C, \neg L \in P\) with \(\langle L, \neg L \rangle\) complementary:

\[
\text{reduction rule} \quad \frac{(C \setminus L, M, P)}{C, M, P}
\]

for some \(L \in C, C_1 \in M, \neg L \in C_1\) with \(\langle L, \neg L \rangle\) complementary:

\[
\text{extension rule} \quad \frac{(C \setminus L, M, P)}{C, M, P}
\]

\[
\frac{(C_1 \setminus \neg L, M \setminus C_1, P \cup \{L\})}{C, M, P}
\]
Representing the Connection Calculus in Prolog
LeanCoP: Connection Calculus as Prolog Program

- Syntax: First order syntax on top of prolog structures
- Calculus: Connection calculus