An Introduction to Prolog Programming

Ulle Endriss
Institute for Logic, Language and Computation
University of Amsterdam

Logic and Prolog

In this lecture we are going to see how Prolog programs can be interpreted as sets of logic formulas. In fact, when processing a query, Prolog is actually applying the rules of a logical deduction system (which is called resolution).

Correspondences

Prolog	**First-order Logic (FOL)**
predicate	predicate
argument	term
variable	universally quantified variable
atom	constant/function/predicate symbol
sequence of subgoals	conjunction
`:-`	implication (other way round)

Question

What is the logical meaning of this program?

```
bigger(elephant, horse).
bigger(horse, donkey).
is_bigger(X, Y) :- bigger(X, Y).
is_bigger(X, Y) :- bigger(X, Z), is_bigger(Z, Y).
```

Answer

The translation of a Prolog program is usually represented as a *set* of formulas, with each formula corresponding to one of the clauses in the program:

$$
\begin{aligned}
\{\quad & bigger(elephant, horse), \\
& bigger(horse, donkey), \\
& \forall x.\forall y.(bigger(x,y) \rightarrow is\_bigger(x,y)), \\
& \forall x.\forall y.\forall z.(bigger(x,z) \wedge is\_bigger(z,y) \rightarrow is\_bigger(x,y)) \quad \}
\end{aligned}
$$

Such a set is to be interpreted as the *conjunction* of all the formulas in the set.

Translation of Programs

- Predicates remain the same (syntactically).
- Commas separating subgoals become $\wedge$.
- `:-` becomes $\rightarrow$ and the order of head and body is changed.
- Every variable is bound by a universal quantifier ($\forall$).

Meaning of Prolog Programs

A Prolog program corresponds to a set of formulas, all of which are assumed to be true. This restricts the range of possible interpretations of the predicate and function symbols appearing in these formulas. The formulas in the translated program may be thought of as the premises in a proof.

If Prolog gives a positive answer to a given query, this means that the translation of the query is a logical consequence of these premises (possibly under the assumption of suitable variable instantiations). If Prolog gives a negative answer, this means that the query cannot be true under the assumption that all of the program formulas are true.

Translation of Queries

Queries are translated like rules; the "empty head" is translated as $\bot$ (falsum). This corresponds to the negation of the goal whose provability we are trying to test when submitting a query to Prolog.

Logically speaking, instead of deriving the goal itself, we try to prove that adding the negation of the goal to the program would lead to a contradiction:

$$\mathcal{P}, (A \rightarrow \bot) \ \vdash \ \bot \quad \text{iff} \quad \mathcal{P} \ \vdash \ A$$

Example

The query

```
?- is_bigger(elephant, X), is_bigger(X, donkey).
```

corresponds to the following first-order formula:

$$\forall x.(is\_bigger(elephant, x) \land is\_bigger(x, donkey) \rightarrow \bot)$$

Horn Formulas

The formulas we get when translating all have the same structure:

$$A_1 \wedge A_2 \wedge \cdots \wedge A_n \rightarrow B$$

Such a formula can be rewritten as follows:

$$\begin{array}{ll} A_1 \wedge A_2 \wedge \cdots \wedge A_n \rightarrow B & \equiv \\ \neg(A_1 \wedge A_2 \wedge \cdots \wedge A_n) \ \vee \ B & \equiv \\ \neg A_1 \vee \neg A_2 \vee \cdots \vee \neg A_n \vee B & \end{array}$$

Hence, formulas obtained from translating Prolog clauses can always be rewritten as disjunctions of literals with at most one positive literal. Such formulas are know as *Horn formulas*.

Resolution

The search tree built up by Prolog when trying to answer a query corresponds to a logic proof using *resolution*, which is a very efficient deduction system for Horn formulas.

A short introduction can be found in the notes; for more details refer to theoretically oriented books on logic programming.

Summary: Logic Foundations

- Prolog programs correspond to sets of first-order logic (Horn) formulas.
- During translation, `:-` becomes an implication (from right to left), commas between subgoals correspond to conjunctions, and all variables need to be universally quantified. Queries become (universally quantified) implications with $\bot$ in the consequent.
- Prolog's search to satisfy a query corresponds to a logical proof. In principle, any deduction calculus could be used. Historically, Prolog is based on resolution, which is particularly suited as it is tailored to Horn formulas.