Lecture 8 (Oct 3)

Lecture outline:

- formal definitions of resolution search tree, and answer
- standard Prolog
- Prolog lists

Last time we saw an example of resolution search tree, and how to compute an answer – here are the formal definitions of the relevant concepts.

Definition 1. *Let P be a logic program, and g be a goal clause. A* resolution search tree *for P and g is a possibly infinite labeled tree T such that:*

1. *The root of T is labeled by g;*
2. *The leafs of T are labeled by either $:-$, or "$fail$";*
3. *Each non-leaf node n of T is labeled by some goal clause $:- t_1, \dots, t_n$, and*
   a. *if t_1 does not unify with any of the heads of clauses in P, then n has one child "$fail$";*
   b. *if $C_1, \dots, C_k$ are the clauses of P whose heads unify with t_1, in order of appearance in P, then n has exactly k children $n_1, \dots, n_k$, where child n_i is labeled with the result of resolution of $:- t_1, \dots, t_n$ with C_i on t_1. The edge $n \to n_i$ is labeled with the m.g.u. of t_1 and the head of C_i.*

Definition 2. *Let P be a logic program, g be a goal clause, and T be a resolution search tree for P and g. An* answer *for P and g is a substitution obtained by the composition of all m.g.u. that label the path from g to $:-$ in T, restricted to the variables of g.*

Standard Prolog

Standard Prolog (or, just *Prolog*)is a logic programming system made into a programming language. Here are the things that are specific to Prolog:

Program

Prolog program is a collection of facts, rules, and also goals, although the goals are used only for "special needs" - we may see some of these later. The syntax of clauses is slightly different:

- Facts are written as, for example, p. (note the dot).
- Rules are written as, for example, $p :- r, s, t$. (note the dot).
- Goals in the program are written as, for example, $:- r, s, t$. (note the dot).

Goal

Goal is given as a command line query, for example $?-p$.

Unification

Prolog does not perform occurs check in unification, so for example X and $f(X)$ do unify. Prolog's operator $=$ is for checking unification of two terms: $t1 = t2$ iff $t1$ unifies with $t2$.

Resolution search tree

Constructed in the depth first manner.

- When the refutation is found, Prolog prints an answer, and waits for users input: Enter means "stop search", Prolog answers "Yes" in this case. ";" means "look for more solutions".
- If the refutation not found (or it was found, but user asked for more, and there's no more), Prolog prints "No".

Extras

Prolog is a programming language, and so has many extras, on top of the logic programming system we described, that make it usable. We will cover some of these:

- Lists
- Arithmetic
- Negation
- Search control via Cut
- Extra-logical predicates (predicates about predicates, program database manipulation, etc)
- System predicates
- Operators

Prolog Lists

List is an ordered sequence of elements (terms), can be of any length. Prolog's notation for a list of terms $t_1, t_2, \ldots, t_n$ is $[t_1, t_2, \ldots, t_n]$. An *empty list*, that is a list with 0 elements, is denoted as [].

Example 3. $[1, 2, 3, 4, 5]$ is a list of 5 elements; $[t(X, Y), g(f(X))]$ is a list of two elements.

Definition 4. *Given a list* $L = [t_1, t_2, \dots, t_n]$ *the* head *of of* L *is the term* t_1*, and the* tail *of* L *is the list* $[t_2, \dots, t_n]$.

Example 5. The head of $[1, 2, 3, 4, 5]$ is 1, the tail is $[2, 3, 4, 5]$.

Lists can be constructed and using operator | which takes two arguments: the first should be a term (note that a list is also a term), and the second is a list. Then, if $L = [l_1, dots, l_k]$, and $t_1, \dots, t_n$ are terms ($n >= 1$),

$$[t_1, \dots, t_n | L]$$

is the list

$$[t_1, \dots, t_n, l_1, \dots, l_k]$$

Example 6.

$$[1|[2, 3, 4]] = [1, 2, 3, 4]$$
$$[f(X), g(Y)|[4, 5, 6]] = [f(X), g(Y), 4, 5, 6]$$

Remember that = in Prolog is unification, so given a query $[H|T] = [1, 2, 3, 4, 5]$ Prolog will answer

$$H = 1$$
$$T = [2, 3, 4, 5]$$

"Internally" lists are represented using a predicate $.(H, T)$, in which H is a term, and T is a list. The operator | is just the "external" notation for . : $[t|L]$ is simply $.(t, L)$, and $[t_1, \dots, t_n|L]$ is simply $.(t_1, .(t_2, \dots, .(t_n, L)))$.

Example 7. The list $[1, 2, 3, 4, 5]$ is represented internally as

$$.(1, .(2, .(3, .(4, .(5, [])))))$$

Thinking in terms of internal representation may help to figure out whether two lists unify.