Logic Programming:
The Computational Model

- Unification

- Resolution
Terms

• Constants

  1. Atoms – fred, +, =/=, ’ABC’

  2. Numbers – 0, 1, 2.345

• Variable – X, Gross_pay,_abc

• Compound term – like(john, mary)
  Consists of a functor and one or more arguments
Unification

Two terms *unify* if there is a substitution that can be made for the variables in the two terms such that the two terms become identical.

\[ f(X, Y, h(3, a)) \] unifies with \[ f(3, a, h(X, Y)) \]

via the substitution \( X = 3, Y = a \). Such a substitution is also called a *unifier*

If two terms \( t_1 \), and \( T_2 \) are unifiable there is a *most general unifier* \( \sigma \) for them and for any unifier \( \rho \) we have \( \rho = \sigma \circ \rho' \) for some \( \rho' \).
A Unification Algorithm

Given a set of equations $t_1 = t_2$ apply in any order one of the following (non exclusive steps)

If there is an equation $f = g$ with $f$ and $g$ distinct constants exit with failure,

If there is an equation $f = g$ with $f$ and $g$ functional terms with distinct principal function symbols or arity then exit with failure,

If there is an equation $f = g$ with one of $f$ a constant and the other a functional term exit with failure,
If there is an equation $f = g$ with $f$ and $g$ functional terms with the same function symbols and arity, add to the equations the equations, $t_1 = s_1, \ldots t_n = s_n$ given by the arguments of $f$ and $g$.

If $f$ and $g$ are both variables, add $f = g$ to the substitution, and apply the substitution.

If either of $f$ and $g$ is a variable which occurs strictly within the other, exit with failure.

If $f$ is a variable, add $f = g$ to the substitution and perform the substitution.

If $g$ is a variable, add $g = f$ to the substitution and perform the substitution.
Unification in Prolog;
The Occurs Check

\(X=Y\) means the \(X\) and \(Y\) can be unified.

Note however the equation \(X = f(X)\)

Normally Prolog does not check for this. But it can be made to.
Horn Clauses

A *Horn Clause* is a disjunction of literals containing at most one positive literal.

- `happy(fred)`
- `\neg passing(fred, calculus)`

Prolog rules can be interpreted as Horn clauses thus $H ::= B_1, \ldots, B_n$ is $H \lor \neg B_1 \lor \ldots \lor \neg B_n$

A fact is a Horn clause that is a single positive literal.
The Basis of resolution

Modus Ponens:

\[ p \Rightarrow q \]

\[ p \]

therefor

\[ q \]

We can rewrite this as

\[ \neg p \lor q \]

\[ p \]
Resolution

Given a goal $G = A_1, \ldots, A_k, \ldots A_n$ and a rule $R$

$B : -B_1, \ldots, B_p$

such that there is a unifier $\sigma$ of $A_k$ and $B$ we can resolve $G$ with $R$ using $\sigma$ to get a goal

$$A_1, \ldots A_{k-1}, B_1\sigma, \ldots, B_p\sigma, A_{k+1}, \ldots A_n$$
Example

Suppose that we have the goal \texttt{grandfather(john,Y)} and the rule

\texttt{grandfather(GX,GZ) :- father(GX,GY), mother(GY,GZ)}

then the substitution

\[ \sigma = \{ \texttt{GX} \leftarrow \texttt{john}, \texttt{Y} \leftarrow \texttt{GZ} \} \]

unifies the head of the rule with the goal and resolution gives a new goal.

\texttt{father(john, GY), mother(GY, Y)}
Soundness and Completeness of Resolution

1. Resolution is **sound**

2. Resolution is **complete** in the sense that if there is solution to the goal resolution will find it. – Careful this as we shall see does not mean the Prolog will.
The Herbrand Domain and Prolog Semantics

The symbols used in a Prolog program $P$ define a language.

$P$ is given by the constants and function symbols that can appear in the program.

$H_P$ is the set of all ground terms that can be formed from these symbols.

Example.

Suppose we have the following Prolog program.

\[
\begin{align*}
\text{int}(0). \\
\text{int}(s(X)) & : \text{int}(X)
\end{align*}
\]

Then $H_P\{0, s(0), s(s(0)), \ldots\}$
Interpretation

A (pure) Prolog program has an interpretation on its Herbrand domain.

Intuitively it is constructed as follows.

1. Start with all the facts in the program. This defines a subset $H_0$ of $H_P$

2. Define $H_{n+1}$ to be the subset of $H_P$ that can be obtained from $H_n$ by using a rule once.

3. Then the interpretation is $\bigcup_n H_n$
The Prolog Execution model

The process of using resolution to find a solution to a goal leaves two choices open:

- The order in which one chooses the sub-goal to resolve on;

- The order in which one chooses the clauses (rules) to resolve with

Prolog chooses the leftmost subgoal; that is $A_1$ in $A_1, \ldots A_n$ and the first rule as it appears in the database.
Examples

path1(X,X).
path1(X,Z) :- a(X,Y), path1(Y,Z).

path2(X,X).
path2(X,Z) :- path2(Y,Z), a(X,Y).

path3(X,Z) :- a(X,Y), path3(Y,Z).
path3(X,X).

path4(X,Z) :- path4(Y,Z), a(X,Y).
path4(X,X).
Following Prolog execution 1.

solve(true).
solve((A, B)) :-
    solve(A), solve(B).
solve(A) :-
    clause(A, B),
    solve(B).
solve(A) :- call(A).
The Debugger

Turn this on using `trace/0` (or in SWI `trace(Goal)` also works).

The following is a summary of some debugger commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ret or c</td>
<td>creep</td>
<td>single-step</td>
</tr>
<tr>
<td>s</td>
<td>skip</td>
<td>skip over current goal</td>
</tr>
</tbody>
</table>

`spy(Pred/Arity)` sets a spy point on a predicate. Use `debug/0` with this.