Resources

- Documentation: (You don’t necessarily need to read. But good for reference when you have questions.)
  http://www.swi-prolog.org/dl-doc.html
Menu “Help -> Online Manual” (HTML files in directory “doc”)

Query Prompt

- query prompt
  ?- (Enter goals after “?-“)
  Example: ?- help(help).

- Load a file with clauses
  ?-[swi('myprogram/example.pl')].
  or
  ?-[swi('myprogram/example')].
  (myprogram must be a subdirectory in the swi-prolog program directory)

User Interaction

?- parent(bob,sam). (a query must end with .)
  true (can be proved)

?- parent(bob,jill).
  fail (cannot prove)

?- parent(bill,X), father(X,sam). (user can use multiple lines to write a query)
  fail

?- parent(X, sam).
  X = jill ;
  X = bob
  (user typed ; to ask for more answers.)
### Debugging

?- trace, parent(X, sam).
Call: (8) parent(_G494, sam) ? creep
Call: (9) mother(_G494, sam) ? creep
Exit: (9) mother(jill, sam) ? creep
Exit: (8) parent(jill, sam) ? creep
X = jill ;
Redo: (8) parent(_G494, sam) ? creep
Call: (9) father(_G494, sam) ? creep
Exit: (9) father(bob, sam) ? creep
Exit: (8) parent(bob, sam) ? creep
X = bob

More details in section 2.9 and 4.2.8 of the manual

### Graphical Debugger

### Prolog Syntax

```
<clause> ::= <fact> | <rule>

<fact> ::= <term> .

<rul> ::= <term> :- <termlist> .

<termlist> ::= <term> | <term> , <termlist>

<term> ::= <variable> | <constant> | <compound-term>

<constant> ::= <number> | <atom>

<compound-term> ::= <atom> ( <termlist> )
```

### Basic Syntax

- `:` for `←`
- `:` for `and`

- Uppercase: variable
- Lowercase: other names (constants, atom [i.e., name of predicate])

- Built-in predicates:
  - `read`, `write`, `nl` (newline)
  - `=<, >, =>, =, , , mod, div`

    (Note it is `=<`, not `<=`)

### Arithmetic

- Arithmetic operation can use prefix or infix notations.
  - `+ (3, 4)`
  - `3+4`

- Value is not immediately evaluated.
  - `? write(3+5).`
  - `7: X is 3+5.` (is a predicate that evaluates 3+5)
  - `x=7.`
  - `7: 3+4 = 4+3. ` (these are two different terms)
  - `fail.`
  - `7: X is 3+4, Y is 4+3, X = Y.` (unification)
  - `X=7, Y=7.`
Unification

- The semantics of = is determined by unification, i.e., = forces unification.

  ?- me = me.
  X = me.
  X = me.

- (See unification algorithm in Page 556)

  ?- me = you.
  ?- me = X.
  Y = a.
  X = f(X).

Unification for List Operations

?- [H|T]=[1,2,3].
  H = 1,
  T = [2,3]

?- [H1,H2|T]=[1,2,3].
  H1 = 1,
  H2 = 2,
  T = [3]

?- [H1,H2,H3|T]=[1,2,3,4,5].
  H1 = 1,
  H2 = 2,
  H3 = 3,
  T = [4,5]

List Operations

- Concatenation:
  ?- X = [0,1|[2,3,4]].
  X = [0,1,2,3,4]

- Get elements, or tail :
  ?- [H1,H2|[3,4]] = [0,1|[2,3,4]]
  What do we get?
  fail.

  ?- [H1,H2|[3,4]] = [0,[1,2],[3,4]]
  What do we get?
  H1 = 0,
  H2 = [1,2].

Define List Operation Predicates

- cons(X,Y,L) :- L = [X|Y].
  ?- cons(0,[1,2,3],A).
  ?- cons(X,[Y,[1,2,3]]).

- Another definition
  append([],Y,L).
  append([H|T],L1,L) :- append(T,L1,L).

- reverse([L]).
  reverse([H|T],L) :- reverse(T,L1), append(L1, [H], L).
Prolog’s Search Strategy

Resolution and Unification

- Order matters:
  - The order to resolve subgoals.
  - The order to use clauses to resolve subgoals.
- Thus programmers must know the orders used by the language implementations, in order to write efficient or even correct program. (Search Strategies)

Prolog’s Strategy

- **Depth-first search**
  - The order to resolve subgoals.
    - (left to right)
  - The order to use clauses to resolve subgoals.
    - (top to bottom)
- **Backtrack:**
  try another clause when it fails.

Example 1

- **Facts:**
  
  \[
  \text{ancestor}(X,Y) :- \text{ancestor}(X,Z), \text{parent}(Z,Y).
  \]
  
  \[
  \text{ancestor}(X,Y) :- \text{parent}(X,Y).
  \]
  
  \[
  \text{parent}(X,Y) :- \text{mother}(X,Y).
  \]
  
  \[
  \text{parent}(X,Y) :- \text{father}(X,Y).
  \]
  
  father(bill,jill).
  mother(jill,sam).
  father(bob,sam).

- **Queries:**
  
  \[
  ?- \text{ancestor}(\text{bill}, \text{sam}).
  \]

Example 1

- **Facts:**
  
  \[
  \text{ancestor}(X,Y) :- \text{ancestor}(X,Z), \text{parent}(Z,Y).
  \]
  
  \[
  \text{ancestor}(X,Y) :- \text{parent}(X,Y).
  \]
  
  \[
  \text{parent}(X,Y) :- \text{mother}(X,Y).
  \]
  
  \[
  \text{parent}(X,Y) :- \text{father}(X,Y).
  \]
  
  father(bill,jill).
  mother(jill,sam).
  father(bob,sam).

- **Queries:**
  
  \[
  ?- \text{ancestor}(\text{bill}, \text{sam}).
  \]
Example 1

- Facts:

  \begin{align*}
  \text{ancestor}(X,Y) & :\text{= ancestor}(X,Z), \text{parent}(Z,Y). \\
  \text{parent}(X,Y) & :\text{= mother}(X,Y). \\
  \text{father}(X,Y) & :\text{= father}(X,Y). \\
  \text{mother}(X,Y) & :\text{= mother}(X,Y). \\
  \text{father}(\text{bill},\text{jill}) & . \\
  \text{mother}(\text{jill},\text{sam}) & . \\
  \text{father}(\text{bob},\text{sam}) & .
  \end{align*}

- Queries:

  \begin{align*}
  \text{ancestor}(\text{bill},\text{Z2}), \text{parent}(\text{Z2},\text{Z1}), \text{parent}(\text{Z1},\text{sam}). \\
  \text{ancestor}(\text{X3},\text{Y3}) & :\text{= ancestor}(\text{X3},\text{Z3}), \text{parent}(\text{Z3},\text{Y3}).
  \end{align*}

Resulting in an infinite loop. Original order was bad.

Example 2

- Facts:

  \begin{align*}
  \text{ancestor}(X,Y) & :\text{= parent}(X,Y). \\
  \text{ancestor}(X,Y) & :\text{= ancestor}(X,Z), \text{parent}(Z,Y). \\
  \text{parent}(X,Y) & :\text{= mother}(X,Y). \\
  \text{parent}(X,Y) & :\text{= father}(X,Y). \\
  \text{father}(\text{bill},\text{jill}) & . \\
  \text{mother}(\text{bill},\text{sam}) & . \\
  \text{father}(\text{bob},\text{sam}) & .
  \end{align*}

- Queries:

  \begin{align*}
  ?= \text{ancestor}(\text{bill},\text{sam}). \\
  \text{What will happen?}
  \end{align*}

Note that we change the order of the first two clauses in facts.

Example 3

- Facts:

  \begin{align*}
  \text{ancestor}(X,Y) & :\text{= parent}(X,Y). \\
  \text{ancestor}(X,Y) & :\text{= ancestor}(X,Z), \text{parent}(Z,Y). \\
  \text{parent}(X,Y) & :\text{= mother}(X,Y). \\
  \text{parent}(X,Y) & :\text{= father}(X,Y). \\
  \text{father}(\text{bill},\text{jill}) & . \\
  \text{mother}(\text{jill},\text{sam}) & . \\
  \text{father}(\text{bob},\text{sam}) & .
  \end{align*}

- Queries:

  \begin{align*}
  ?= \text{ancestor}(\text{X},\text{bob}).
  \end{align*}

Resulting in an infinite loop. Original order was bad.
Example 4

• Facts:

\[
\begin{align*}
\text{ancestor}(X,Y) & :- \text{parent}(X,Y). \\
\text{ancestor}(X,Y) & :- \text{parent}(X,Z), \text{ancestor}(Z,Y). \\
\text{parent}(X,Y) & :- \text{mother}(X,Y). \\
\text{parent}(X,Y) & :- \text{father}(X,Y). \\
\text{mother}(jill,sam). \\
\text{father}(bob,sam). \\
\end{align*}
\]

• Queries:

\[ ?- \text{ancestor}(X,bob). \]

What will happen?

Note that we change the order of the two subgoals in clause (2).