Prolog Notes

<u>INSTANTIATION</u>: binding of a variable to value (and thus, a type)

<u>UNIFICATION</u>: Process of finding an instantiation of a variable for which "match" is found in database of facts and rules

Instantiation & Unification

FACTS
- **(assert (color (apple, red))).**
- **(assert (color (banana, yellow))).**

color (X, yellow).

Ask the question (goal):
Does there exist (or, Give me) an X such that X is the color yellow

X = apple **color (apple, yellow)**
instantiation **no matching pattern**

X = banana **color (banana, yellow)**
instantiation **match**

X = banana **results in match of goal with database item**

Prolog Notes

- DISJUNCTIVE RULES: X if Y or Z

```
(assert ((parent(X, Y) :-  father(X, Y)))).
(assert ((parent(X, Y) :-  mother(X, Y)))).
```

or

```
(assert ((parent(X, Y) :-  father(X, Y);
  mother(X, Y)))).
```

Prolog Notes

- CONJUNCTIVE RULES: X if Y AND Z

```
(assert((father(X, Y)  :- parent(X, Y),
male(X)))).
```

- NEGATION RULES: X if Not Y

```
(assert((good(X) :- not(bad(X))))).
(assert((mother(X, Y) :- parent(X, Y),
not(male(X))))).
```

“Older” Example

```
older(george, john).
older(alice, george).
older(john, mary).
older(X, Z) :- older(X, Y), older(Y, Z).
```

- When we ask a query that will result in TRUE, we get the right answer:

```
?- older(george, mary).
yes
```

- When we ask a query that will result in FALSE, we get into an endless loop

```
?- older(mary, john).
```

Left Recursion Problem

- The first element in older is the predicate that is repeatedly tried
- To solve the problem, remove the older rule and replace with:

```
is_older(X, Y) :- older(X, Y).
is_older(X, Z) :- older(X, Y),
is_older(Y, Z).
```

- Now:

```
?- is_older(mary, john).
false
```

Prolog Notes

- Prolog is more than "LOGIC"
  - Math
  - List manipulation

Consult File Format

[x]. or consult(x).

- File x.pl:

```
husband(tommy, claudia).
husband(mike, effie).
mother(claudia, sannon).
mother(effie, jamie).
father(X, Y) :- mother(W, Y), husband(X, W).
parent(X, Y) :-father(X, Y); mother(X, Y).
```

- Note: No assert's, but can still state **Facts and Rules**

Consult File

- Cannot state question/goal in a consult file

```
| ?- consult(x).
```

Suggested Approach to Specifying Solution

- Use a consult file to define facts and rules
  - Instantiate prolog
  - "consult" file interactively
  - Interactively ask questions to see if facts/ rules yield expected results
  - Change consult as needed
    - Need to reinitiate prolog and re"consult"

Suggested Approach to Specifying Solution

- Construct I/O redirected file to include
  - Consult file and queries, e.g.,
    swipl < input.fle
  - You may use ";" to ask "Is there another answer?"
    - The initial query CANNOT have anything on the line after the ".", and
    - There must be a blank line after ";"

input.fle

```
consult(cnslt).
query1.
;

query2.
```

SWI-Prolog: Access & Nuance

- SWI-Prolog on Rlogin is located in the directory:
  - /home/staff/arthur/bin/swipl
- swipl prints output to STDERR (file descriptor 2). To redirect output to a file you must precede ">" with a "2" :
  - swipl < input.fle 2> output.fle

Prolog – Issues/Limitations

- "Closed World"
  - the only truth is that known to the system
- Efficiency
  - theorem proving can be extremely time consuming
- Resolution order control
  - Prolog always starts with left side of a goal, and always searches database from the top. Have some control by choice of order in the propositions and by structuring database.

Prolog – Issues/Limitations

- Prolog uses backward chaining (start with goal and attempt to find sequence of propositions that leads to facts in the database).
- In some cases forward chaining (start with facts in the database and attempt to find a sequence of propositions that leads to the goal) can be more efficient.
- Prolog always searches depth-first, though breadth-first can work better in some cases.

Prolog – Issues/Limitations

- The Negation Problem -- failure to prove is not equivalent to a logical not
  - not(not(some_goal)) is not necessarily equivalent to some_goal