1 MA410 Artificial Intelligence - Prolog Introductory Lab

1.1 Programming Language Categories

Imperative e.g. C, Maple, Java, Basic

Functional e.g. Lisp, ML, Haskell

Logic Declarative e.g. Prolog

Imperative/Procedural languages tell a computer "how to solve a problem". (What to do)

Declarative says "what problem we want solved". (What is true) We give clues to the solution.

1.2 Basics of Prolog

Prolog programs are about relations between entities. We give a database of pieces of information:

- *facts* - always true
- *rules* - dependent on others

To run a Prolog program, we ask questions about the information in the database.

1.3 Getting started

- Open website http://www.maths.nuigalway.ie/∼barry & download `lab0.pl` into appropriate directory.
- Open this file with some text editor e.g. notepad.

1.3.1 Loading prolog

Windows: Double-click icon marked "prolog".

Linux: Go to `http://www.swi-prolog.org` and install.

1.3.2 Prolog program and interpreter

- The file `lab0.pl` is used for storing facts & rules.
- Prolog interpreter is for running queries.
- To load file type '`?- consult('*lab0.pl').`' in prolog where * stands for file path.

1.4 Writing facts

The examples below are illustrative. Please try to construct your own examples.

Example 1 (Single Fact)
Record the fact *"it is raining"*. This can be done by editing `lab0.pl` and simply typing

```
it_is_raining.
```

Now go into the prolog interpreter and query

```
?- it_is_raining.
```

then try `?- not(it_is_raining).`

Example 2 (Fact with single argument)
Record the facts *"Mary is female"*, *"John is male"*:

```
female(mary).   male(john).
```

Use the prolog interpreter to query

```
?- female(mary).  % Is Mary female?
?- female(X).     % Who are female?
```

The uppercase `X` in the above example refers to a variable. Remember uppercase means variable, lowercase means atom.

Example 3 (Fact with single relation)
Record the fact "Mary is a parent of John":

```
parent(mary, john).
```

Use the prolog interpreter to query

```
?- parent(mary, john).  % Is Mary a parent of John?
?- parent(X, john).     % Who is a parent of John?
?- parent(mary, X).     % Who has Mary as a parent?
?- parent(X, Y).        % Who are parents and of whom?
```

Example 4 (Defining a rule)
Record the rule *"If someone is a parent of x and female then they are a mother of x."*:

```
mother(X,Y) :- female(X), parent(X,Y).
```

Use the prolog interpreter to query

```
?- mother(X, john).  % Who is a mother of John?
```

Example 4 (Use of '_' character)
The underscore character '_' is used in place of a variable when we don't want a response for that variable. E.g. We can query the database to find out all those who are mothers: `?- mother(X, _).`

Example 5 (Recursion)
Record the rule that
"an ancestor is a parent or a parent of an ancestor":

```
ancestor(X,Y) :- parent(X,Y).
ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).
```

Example 6 (Lists)
Prolog Lists are enclosed within square brackets, e.g. `[a,b,c]`. Lists themselves can contain sublists which in turn can contain their own sublists etc.

List example: `[a, b, [c,d]]`.

Can also be written in form `[Head | Tail]` notation as `[a | [b, [c,d]]]` where Head contains a single element and Tail the remainder of the list.

Write a prolog *predicate* `del(X, L1, L2)` that deletes `X` from `L1` to return `L2`:

```
del(X, [X|T], T).
del(X, [Y|T], [Y|T1]) :- del(X, T, T1).
```