Knowledge Representation Laboratory 2: Recursive Arithmetic in Prolog

CITS3005

This laboratory will explore some of the basic concepts of Prolog programming through a recursive implmentation of arithmetic.

1 Setting up Prolog Environment

By now you should have had a chance to try a number of different prolog environments, including Gnu-Prolog, SWI-prolog, web environments and others. Finding a good editor is useful. Some packages some with their own editor, or you can configure editors like VSCode to do syntax highlighting for prolog (make sure it is not interpreting `file.pl` as a perl file).

If you find a good development environment that your happy with, please share it with the class through MS Teams.

2 Continue tutorial...

Continue with the TutorialsPoint Prolog Tutorial, up to page 10: *Conjunctions and Disjunctions.*

3 Recursive Arithmetic

Build a Prolog program to do recursive arithmetic. It should have:

1. a constant, `zero`, to represent zero.
2. a function, `next(X)`, that represents the *next number* (so `next(zero)` represents one).
3. a predicate, `sum(X,Y,Z)`, which is true if $X + Y = Z$.
4. a predicate, `mult(X,Y,Z)` which is true if $X \times Y = Z$.
5. a predicate, `equals(X,Y)` which is true if $X = Y$.
6. a predicate, `lessThan(X,Y)` which is true if $X < Y$¿

Test your program and consider the efficinecy of the implementations. Is there a way to get faster annswers? Next, add additional functions:

1. Write a program `binary(X)` that will print the binary representation of X out, so for example, `binary(next(next(zero))` will print out `10`.
2. Implement predicates for odd, even and prime numbers.

4 Extension

Finally, consider how you would extend your arithmetic program to handle negative numbers (so you have `before(zero)`, and `subtract` etc), or modular arithmetic (so for example if all operations are modular 5, `mult(next(next(next(zero))), next(next(zero)), next(zero))` is true.