Assignment 6:
Prolog

15-317: Constructive Logic

Out: Thursday, October 16, 2008
Due: Thursday, October 23, 2008, before class

1 Division (10 Points)

Here is code for addition and multiplication:

```
nat(z).
nat(s(N)) :- nat(N).

plus(z, N, N).
plus(s(M), N, s(P)) :- plus(M, N, P).

times(z, _, z).
times(s(N1), N2, N3_2) :-
times(N1, N2, N3),
plus(N3, N2, N3_2).
```

Task 1 (10 pts). Determine if the `times` predicate can be used to calculate exact division: Given `m` and `n`, find a `q` such that `m = (n * q)`, and fail if no such `q` exists.

If so, state the mode with which `times` is invoked to compute exact division, and argue that `times` has that mode.

If not, give counterexamples for the different modes with which `times` could be invoked to compute exact division, and write another program `exactDiv(m,n,q)` to perform exact divison.

You may answer the written portion of this question in comments in your prolog file (the line comment character is `%`) or in a separate written handin.

2 Unary to binary and back (10 Points)

We can represent a binary numbers as a list of bits, where 0 is represented by the empty list, and the most-significant bit is at the end of list. This most-significant bit (the last element of the list) must be one: no trailing zeros are permitted. For example, 2 is represented by the list `[zz,oo]`, where `zz` is the zero bit and `oo` is the one bit.

Here are predicates recnogizing bits and binary numbers:

```
bit(zz).  %% the zero bit
bit(oo).  %% the one bit

bitlist([]).
bitlist([H|T]) :- bit(H),bitlist(T).

endsWithOO([oo]).
endsWithOO([_|T]) :- endsWithOO(T).

binaryNumber([]).
binaryNumber(L) :- bitlist(L),endsWithOO(L).
```

Task 1, code (7 pts). Define two relations to convert back and forth between unary and binary:

```
unaryToBinary(U,B)   mode +U -B
binaryToUnary(U,B)   mode -U +B
```

Task 2, written (3 pts). How do your two relations differ? Can you use the same relation with both modes? You may answer this question in comments in your prolog file or in a separate written handin.

3 Mergesort (10 points)

Write a relation

```
mergesort(Un, Sorted)   mode +Un -Sorted
```

that sorts a list of integers into increasing order using mergesort. Given any unsorted list `Un`, this predicate must compute a list `Sorted` containing the same elements in increasing order. Use the following comparision predicates:

```
H1 =< H2
H1 > H2
```

Hint: you should define auxiliary predicates:

```
partition(In, FirstHalf, SecondHalf)  mode +In -FirstHalf -SecondHalf
merge(L1, L2, L3)                     mode +L1 +L2 -L3
```

The first partitions a given list `In` into two halves, and the second merges two sorted lists into a single sorted list containing all of their elements.

4 Dutch National Flag (10 points)

The Dutch national flag problem is to take a list of elements that are either red, white, or blue and return a list with all red elements first, followed by all white elements, with all blue elements last (the order in which these colors appear on the Dutch national flag). We represent the property of being red, white, or blue with three predicates, `red(x)`, `white(x)`, and `blue(x)`. You may assume that every element of the input list satisfies exactly one of these three predicates.

Write a Prolog program

```
dutchflag(L1,L2)    mode +L1 -L2
```

to solve the Dutch national flag problem. Try to take advantage of the intrinsic expressive power of logic programming to obtain an elegant program.

5 Running Prolog / Handin Instructions

- GNU Prolog is installed in

  ```
  /afs/andrew/course/15/317/bin/gprolog
  ```

  To run it, you must have this `bin` directory in your path.

  In `tcsh`, do this:

  ```
  setenv PATH "/afs/andrew/course/15/317/bin:$PATH"
  ```

  In `bash`, do this:

  ```
  export PATH="/afs/andrew/course/15/317/bin:$PATH"
  ```

  You may want to add this to your startup files so that you don't have to reset your path every time you want to run Prolog.

  Once you run Prolog, load your file using the `consult` command:

  ```
  | ?- consult('yourfile.pl').
  ```

  Then you can run queries:

  ```
  | ?- mergesort([2,4,6,5,1,3],Ls).

  Ls = [1,2,3,4,5,6] ?
  ```

- To hand in your code, copy a file `hw06.pl` to your handin directory:

  ```
  /afs/andrew/course/15/317/submit/<yourid>/hw06.pl
  ```