Programming Paradigms Fall 2022 — Problem Sets

by Nikolai Kudasov

November 3, 2022

1 Problem set №10

Consider the following knowledge base:

```
% student(Name, Group)
student(alisa,  2).
student(bob,    1).
student(chloe,  2).
student(denise, 1).
student(edward, 2).

% friend(Name, Name)
friend(alisa, bob).
friend(alisa, denise).
friend(bob, chloe).
friend(bob, edward).
friend(chloe, denise).
friend(denise, edward).

% parent(Parent, Child)
parent(marjorie, bart).
parent(marjorie, lisa).
parent(marjorie, maggie).
parent(homer, bart).
parent(homer, lisa).
parent(homer, maggie).
parent(abraham, homer).
parent(mona, homer).
parent(jacqueline, marjorie).
parent(jacqueline, patty).
parent(jacqueline, selma).
parent(clancy, marjorie).
parent(clancy, patty).
parent(clancy, selma).
parent(selma, ling).

% unary(Number)
unary(z).
unary(s(X)) :- unary(X).
```

1. Draw search trees for the following queries:

   (a) `?- friend(alisa, Y), friend(Y, Z).`

   (b) `?- friend(X, Y).`

   (c) `?- parent(jacqueline, Y), parent(Y, ling).`

2. Write down predicate groupmates/2 that checks whether two students are from the same group.

```
?- groupmates(alisa, bob)
false

?- groupmates(alisa, edward)
true
```

3. Implement predicate relative/2 that checks whether two people are related by blood (share a common ancestor):

```
?- relative(selma, patty)
true

?- relative(lisa, ling)
true

?- relative(lisa, selma)
true

?- relative(homer, selma)
false
```

4. Implement the following predicates for unary numbers:

   (a) Implement a predicate double/2 that checks if first number is exactly two times the second:

   ```
   ?- double(s(s(z)), s(s(s(s(z)))))
   true

   ?- double(s(s(z)), X)
   X = s(s(s(s(z))))

   ?- double(X, s(s(s(s(z)))))
   X = s(s(z))

   ?- double(X, s(s(s(z))))
   false
   ```

   (b) Implement a predicate leq/2 that checks if the first number is less than or equal to the second numbers:

   ```
   ?- leq(s(s(z)), s(s(s(z))))
   true

   ?- leq(s(s(s(s(z)))), s(s(s(z))))
   false
   ```

   (c) Implement multiplication for unary numbers as a predicate mult/2:

   ```
   ?- mult(s(s(z)), s(s(s(z))), X)
   X = s(s(s(s(s(s(z))))))

   ?- mult(X, s(s(s(z))), s(s(s(s(s(s(z)))))))
   X = s(s(z))
   ```

(d) Implement a predicate powerOf2/2 such that powerOf2(N, M) is true when M is equal to 2 to the power of N:

```
?- powerOf2(s(s(z)), s(s(s(s(z)))))
true

?- powerOf2(X, s(s(s(z))))
false

?- powerOf2(s(z), X)
X = s(s(z))

?- powerOf2(X, Y)
X = z, Y = s(z)
X = s(z), Y = s(s(z))
X = s(s(z)), Y = s(s(s(s(z))))
...
```

Hint: for the last query to produce each result in finite time, you need to put an upper bound on the second argument, e.g. using leq/2.