CPSC 449 – Programming Paradigms

Assignment #4

Due Date: Friday, December 07, 11:59 PM

Individual Work:

All assignments in this course are individual work. Students are advised to read the guidelines for avoiding plagiarism available on the course outline. Students are also advised that electronic tools such as MOSS may be used to detect plagiarism.

Late Penalty:

Late submissions will not be accepted.

Submissions:

Submit `assignment04.pl` file that contains the required predicates in `D2L`.

Make sure your assignment is working on GNU-Prolog (or lab machines). Your TAs would use lab computers to evaluate your assignment.

Sample Run:

We have not provided sample run as the questions self explanatory (I believe). A sample solution file may be uploaded in D2L lately.

Problem Description:

1- **Family relationship:** For the family relationship, consider the fact of the form `mother(name1, name2)`, where name1 is the mother of name2:

- Define a predicate `sister(X, Y)`, which holds iff X and Y are sisters.
- Define a predicate `cousin(X, Y)`, which holds iff X and Y are cousins.
- Define a predicate `granddaughter(X, Y)`, which holds iff X is a granddaughter of Y.
- Define a predicate `descendent(X, Y)`, which holds iff X is a descendent of Y.

You can create your own knowledge base based on the form of the facts provided above.

For Q2 – Q7:

Consider the representation of a set as a list that does not have any duplicated elements:

Define the following predicates:

2- `member(X,L)`, which holds iff the element X occurs in L.
3- `subset(L,K)`, which holds iff L is a subset of K.
4- `disjoint(L,K)`, which holds iff L and K are disjoint (i.e. they have no elements in common).
5- `union(L,K,M)`, which holds iff M is the union of L and K.
6- `intersection(L,K,M)`, which holds iff M is the intersection of L and K.
7- `difference(L,K,M)`, which holds iff M is the difference of L and K.

8- Define a predicate `occurrences(X,L,N)` which holds iff the element X occurs N times in the list L.
9- Define a predicate `quicksort(L,K)` which, given a list of integers L, returns an ordered list K obtained from L with the method of quicksort in ascending order.

10- Consider the following graph (bi-directional) with edges as following facts.

```
edge(1,2). % read as there is an edge from 1 to 2
edge(1,4).
edge(1,3).
edge(2,3).
edge(2,5).
edge(3,4).
edge(3,5).
edge(4,5).
```

Define a predicate path/3 that generates all the paths between two edges.

Additional Challenge:

For the additional challenge, consider the following edges with length. Find the shorted path between two edges.

Below are the edges as facts with lengths:

```
edge(1,2,1).
edge(1,4,3.5).
edge(1,3,2.5).
edge(2,3,1).
edge(2,5,2.5).
edge(3,4,1).
edge(3,5,2.2).
edge(4,5,1).
```

Grading:

A+: All questions of the assignments are completed successfully, included the additional challenge.

A: All questions (1 - 10) except for additional challenge are completed successfully.

B+: First 8 - 9 questions are completed successfully.

B: First 6 - 7 questions are completed successfully.

C: First 4 - 5 questions are implemented successfully.

D / F: Submissions that do not meet the standard for a C will be awarded a grade of D+, D or F depending on functionality, quality and quantity of the submitted code.

Credits:

The text in this assignment is adapted from the course offered by Catuscia Palamidessi.