First-Order Logic: Prolog

Due Dec 7, 2020 by 11:59pm **Points** 12 **Submitting** a file upload

This assignment will use First Order Logic in the form of prolog programs in order to represent knowledge.

Prolog is a language that you can install on your home computer, or you can build small programs online at: **https://swish.swi-prolog.org/ (https://swish.swi-prolog.org/)** which has an interactive online IDE.

Using prolog, we'll build a few different knowledge bases and see if we're able to resolve queries about them.

Kinship Domain

Let's redo the kinship predicates that we talked about in lecture.

Start with the following knowledge:

```
parent(charles, william).
parent(charles, harry).
parent(elizabeth, charles).
parent(george, elizabeth).
parent(george, margaret).
parent(elizabeth, anne).
parent(elizabeth, andrew).
parent(elizabeth, edward).
parent(anne, peter).
parent(anne, zara).
parent(andrew, beatrice).
parent(andrew, eugenie).
parent(edward, louise).
parent(edward, james).
```

(It's like I've been watching the most recent season of The Crown or something.)

Encode into your program the following knowledge:

- The definition of **child** in terms of the **parent** predicate.
- The definition of **sibling** in terms of the **parent** predicate.
  - When finished, the query **sibling(anne, X).** should resolve to:
    X = charles
    X = andrew
    X = edward
- The definition of **cousin** in terms of other predicates (assuming we mean specifically first cousin, not anything removed, etc.)
  - When finished, the query **cousin(beatrice, X)**. should resolve to:

X = william
X = harry
X = peter
X = zara
X = louise
X = james

- The definition of **ancestor** in terms of other predicates.
  - When finished the query **ancestor(X, louise).** should resolve to:
    X = edward
    X = george
    X = elizabeth
  - ... and not result in an infinite loop!

Lists and Sorting

In prolog, lists are defined using square brackets, and the pipe character | is used to separate out the head and tail of a list. A very common construction in prolog predicates is [H | T], which binds H to the first item and T to be a list that is the remaining items (Head and Tail.) Another common construction is using the _ character as a variable, if we don't care about the value of said variable.

Thus, the adding the following to a KB:

```
len([], 0).
len([_ | T], N) :- len(T, M), N is M+1.
```

Adds a recursive definition of length for lists. As a ground fact, an empty list has length 0.

The recursive definition is that a list can be bound to [_, T] which separates it into the variable _, which is the first item, and variable T, which is the rest of the list. Then, that is true if the tail has length M and there is another variable N that is M+1.

As you solve this problem, do not use any built-in procedures.

1. Using these tools, write a prolog clauses that define the predicate **sorted(L)** which is true if and only if list L is in ascending sorted order.
2. Write prolog clauses for the predicate **perm(L, M)** which is true if and only if L is a permutation of M.
3. Define **mysort(L, M)** where M is a sorted version of L using **perm** and **sorted**.
4. Run **mysort** on longer versions of L until you lost patience. What is the time complexity of your algorithm?

FOL Rubric

Criteria	Ratings		Pts
sorted(L)	**3 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
perm(L,M)	**3 pts** **Full Marks**	**0 pts** **No Marks**	3 pts
mysort(L,M) function (2 points) + run (2 points) + time complexity (2 points)	**6 pts** **Full Marks**	**0 pts** **No Marks**	6 pts
			Total Points: 12