Gebze Technical University

Programming Languages
CSE 341

Homework 4
Report

Elif Akgün
1801042251

Part 1

First I wrote all facts as the given graph like flight(istanbul,izmir). Then I wrote rules below.

```
route(X,Y) :- flight(X,Y).
route(X, Y) :- findPath(X, Y, []).
findPath(X, Y, Path) :- flight(Z,X), not(member(Z, Path)), (findPath(Z, Y, [X|Path]); Y = Z).
```

route(X,Y) indicates if there is a path between two cities. It uses helper predicate and it check if there is a path recursively.

You can check if there is flight between to city like flight(city1, city2). Also you can check if there is a path between two city as route(city1, city2). There are some tests below.

```
?- flight(isparta, burdur).
true .

?- flight(istanbul, burdur).
false.

?- route(istanbul, burdur).
true .

?- route(rize, erzincan).
false.
```

```
?- route(edirne,X).
X = edremit ;
X = erzincan ;
X = edremit.
```

```
?- forall(route(edirne,X),writeln(X)).
edremit
erzincan
edremit
true.
```

- To print all connected cities you can use ; or forall.

Part 2

In this part, I add distances between cities. To calculate distance I used given link in the homework file. So new facts like this now: distance(istanbul,izmir,328.80). distance fact shows us distance between two cities. I wrote new rules for this part to calculate distance.

It returns sum of distances in a path. For example there is no direct flight between Istanbul and burdur but there is flight between Istanbul and izmir, izmir and isparta, isparta and burdur. So there is a route between istanbul and burdur.
You can see some tests below.

```
sroute(X,Y,Dist) :- path(X,Y,Dist,[]).
path(X,Y,Dist,Z) :- distance(X,Y,Dist), not(X=Y).
path(X,Y,Dist,Z) :- distance(X,T,D1), not(member(X,Z)), not(T=X), path(T,Y,D2,[X|Z]), Dist is D1 + D2.
```

```
?- sroute(istanbul,burdur,X).
X = 661.95 .

?- sroute(istanbul,edirne,X).
false.

?- sroute(rize,van,X).
X = 373.01 .

?- sroute(edremit,erzincan,X).
X = 1066.26 .
```

Part 3

```
when(102, 10).
when(108, 12).
when(341, 14).
when(455, 16).
when(452, 17).

where(102, z23).
where(108, z11).
where(341, z06).
where(455, 207).
where(452, 207).

enroll(a,102).
enroll(a,108).
enroll(b,102).
enroll(c,108).
enroll(d,341).
enroll(e,455).
```

For this part, I wrote some facts.

- when(X,Y) indicates time of the course X is Y
- where(X,Y) indicates place of the course X is Y
- enroll(X,Y) indicates student X is enrolled in course Y

3.1

I wrote a predicate schedule(S,P,T) that associates a student to a place and time of class. You can see how I defined predicate and some tests below.

```
schedule(S,P,T) :- enroll(S,C), where(C,P), when(C,T).
```

```
?- schedule(a,P,T).
P = z23,
T = 10 ;
P = z11,
T = 12.

?- schedule(b,P,T).
P = z23,
T = 10.

?- schedule(x,P,T).
false.
```

3.2

I wrote another predicate “usage(P,T)” that gives the usage times of a classroom. You can see how I defined predicate and some tests below.

```
usage(P,T) :- where(C,P), when(C,T).
```

```
?- usage(z23,T).
T = 10.

?- usage(z11,T).
T = 12.

?- usage(207,T).
T = 16 ;
T = 17.

?- usage(1,T).
false.
```

3.3

I wrote another predicate “conflict(X,Y)” that gives true if X and Y conflicts due to classroom or time. You can see how I defined predicate and some tests below.

```
conflict(X,Y) :- where(X,P1), where(Y,P2), P1=P2; when(X,T1), when(Y,T2), T1=T2.
```

```
?- conflict(455,452).
true .

?- conflict(455,341).
false.
```

3.4

I wrote another predicate “meet(X,Y)” that gives true if student X and student Y are present in the same classroom at the same time. You can see how I defined predicate and some tests below.

```
meet(X,Y):- enroll(X,C1), enroll(Y,C2), C1==C2, X\=Y.
```

```
?- meet(a,b).
true .

?- meet(a,c).
true.

?- meet(a,e).
false.
```

Part 4

4.1

I wrote predicate “element(E,S)” that returns true if E is in S. You can see how I defined predicate and some tests below.

```
element(E,S) :- member(E,S).
```

```
?- element(1, [1,2,3]).
true .

?- element(4, [1,2,3]).
false.

?- element(X, [1,2,3]).
X = 1 ;
X = 2 ;
X = 3.
```

4.2

I wrote predicate “union(S1,S2,S3)” that returns true if S3 is the union of S1 and S2. I wrotee some helper predicate to check sets. You can see how I defined predicate and some tests below.

```
%union
union(S1,S2,S3) :- unionSet(S1,S2,S3).
unionSet(S1,S2,S3) :- union2(S1,S3), union2(S2,S3), isUnion(S1,S2,S3).
union2(S1,S2) :- foreach(element(X,S1), element(X,S2)).
isUnion(S1,S2,S3):- foreach(element(X,S3), element(X,S1); element(X,S2)).
```

```
?- union([1,2],[3,4],[5,6]).
false.

?- union([1,2],[3,4],[1,2,3,4]).
true.

?- union([1,2],[1,2,3,4],[1,2,3,4]).
true.
```

4.3

I wrote a predicate “intersect(S1,S2,S3)” that returns true if S3 is the intersection of of S1 and S2. I wrotee some helper predicate to check sets. You can see how I defined predicate and some tests below.

```
%intersect
intersect(S1,S2,S3) :- isIntersect(S1,S2,S3).
isIntersect(S1,S2,S3) :- intersect2(S1,S2,X), equivalent(X,S3).
intersect2([],[],[]) :- true,!.
intersect2([],_,[]) :- true,!.
intersect2(_,[],[]) :-true,!.
intersect2([E|S1], S2, [E|S3]):- element(E, S2), !, intersect2(S1, S2, S3).
intersect2([_|S1], S2, S3):- intersect2(S1, S2, S3).
```

```
?- intersect([1,2,3],[3,4,5],[3]).
true.

?- intersect([1,2,3,4],[4,3,5],[4,3]).
true .

?- intersect([1,2,3,4],[4,3,5],[4,6,3]).
false.
```

4.4

I wrote predicate “equivalent(S1,S2)” that returns true if S1 and S2 are equivalent sets. . I wrotee some helper predicate to check sets. You can see how I defined predicate and some tests below.

```
%equivalent
equivalent(S1, S2) :- isEquivalent(S1, S2), isEquivalent(S2, S1).
isEquivalent([],_) :- true, !.
isEquivalent([E|S1],S2):- isEquivalent(S1,S2), element(E,S2).
```

```
?- equivalent([1,2,3],[1,2,3]).
true .

?- equivalent([1,2,3],[3,1,2]).
true .

?- equivalent([1,2,3],[3,1,2,4]).
false.
```

Part 5

Firstly, I create all possible sets with operators using given set. For example [1,2,3,4] can be:

[1,+,2,-,3]

[1,+,2,*,3]

[1,+,2,/,3]

…..

Then, I calculate all possible expressions . If there is an equation, program returns it, otherwise result will be false.

I could not read and write file in this part. I printed the result to terminal.

I wrote some predicates that deletes the last element of list, creates all possible expressions, and does operations, calculates operations.

There are some tests below.

```
For help, use ?- help(Topic). or ?- apropos(Word).

?- list([1,2,3,4]).
false.

?- list([1,2,3,6]).
[1,+,2,+,3]
[1,*,2,*,3]
true.

?- list([1,2,3,7]).
[1,+,2,*,3]
true.

?- list([1,2,3,4,10]).
[1,+,2,+,3,+,4]
[1,*,2,*,3,+,4]
true.

?- list([1,2,3,4,11]).
[1,+,2,*,3,+,4]
[1,-,2,+,3,*,4]
[1,-,2,-,3,*,4]
true.

?- list([4,2,5,3,23]).
[4,*,2,+,5,*,3]
true.

?- list([18,7,10,32,331]).
[18,-,7,+,10,*,32]
[18,-,7,-,10,*,32]
true.
```