Programmeringsspråk TDT4165 - Assignment 5

Ola Kristoffer Hoff

18. November 2022

Task 1: Constraint programming

```
1 payment(0, []).
2 payment(Sum, [coin(Needed, Value, Stock)|Tail]) :-
3     Needed in 0..Stock,
4
5     Sum #= Needed * Value + Rest,
6     payment(Rest, Tail).
```

Figure 1: *Prolog* code for solving the coin problem.

```
?- payment(25,
[coin(Ones,1,11),coin(Fives,5,4),coin(Tens,10,3),coin(Twenties,
20,2)]).
```

```
Fives in 0..4,
_3506#=5*Fives+_3514,
_3506 in 14..25,
Ones+_3506#=25,
Ones in 0..11,
_1260 in 0..25,
_1260#=10*Tens+_1278,
Tens in 0..2,
_1278 in 0..20,
_1278#=20*Twenties,
Twenties in 0..1
```

Figure 2: The results from running the code in figure 1.

```
?- payment(25,
[coin(Ones,1,11),coin(Fives,5,4),coin(Tens,10,3),coin(Twenties,
20,2)]), label([Ones, Fives, Tens, Twenties]).
```

Ones	Fives	Tens	Twenties	
0	1	0	1	1
0	1	2	0	2
0	3	1	0	3
5	0	0	1	4
5	0	2	0	5
5	2	1	0	6
5	4	0	0	7
10	1	1	0	8
10	3	0	0	9

Figure 3: The results from running the code in figure 1 with the printing.

Task 2: Relational programming

Task 2.1: Create a planner

```
15 plan_visited(Cabin, Cabin, Path, TotalDistance, _) :-
16     Path = [Cabin],
17     TotalDistance is 0.
18
19 plan_visited(Cabin1, Cabin2, Path, TotalDistance, Visited) :-
20     not(Cabin1 = Cabin2),
21     not(member(Cabin1, Visited)),
22     distance(Cabin1, CabinNext, D1, 1),
23     plan_visited(CabinNext, Cabin2, TailPath, D2, [Cabin1 | Visited]),
24     \+ member(Cabin1, TailPath),
25     Path = [Cabin1|TailPath],
26     TotalDistance is D1 + D2.
27
28 plan(Cabin1, Cabin2, Path, TotalDistance) :-
29     plan_visited(Cabin1, Cabin2, Path, TotalDistance, []).
```

Figure 4: Code for solving task 2.1.

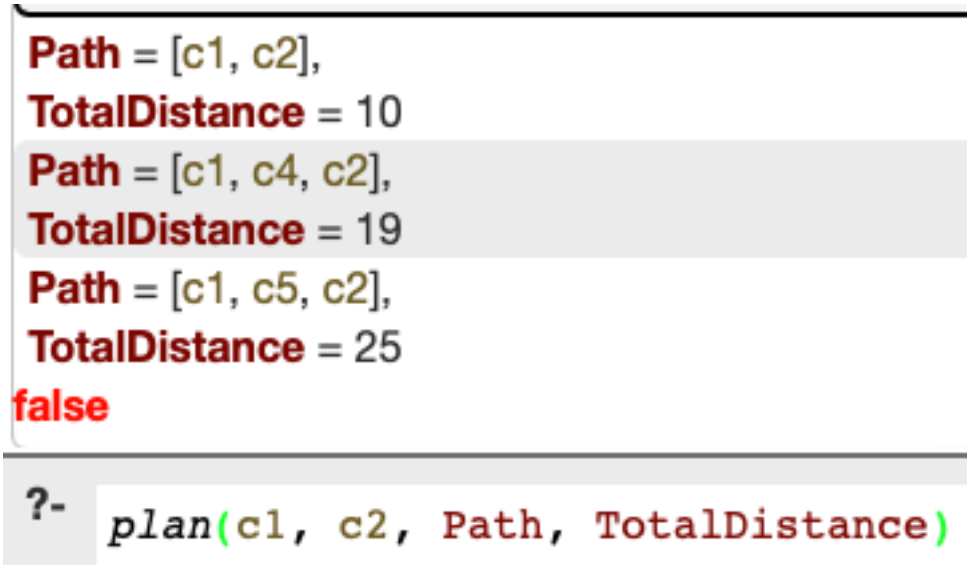

Figure 5: Results after running the code in figure 4.

Task 2.2: Create the planner for the shortest path

I used the previous solution and bound all the possible solutions to a list which I then found the minimum element from.

```
31 bestplan(Cabin1, Cabin2, Path, Distance) :-
32     findallplans(Plans, Cabin1, Cabin2),
33     min_in_list(Plans, (Path, Distance))
34     .
35
36 findallplans(Plans, C1, C2) :-
37     findall([P, D], plan(C1, C2, P, D), Plans).
38
39 min_in_list([[P,D]],(P,D)).
40
41 min_in_list([[P0,D0],[_,D1]|T],(P,D)) :-
42     D0 =< D1,
43     min_in_list([[P0, D0]|T],(P,D)).
44
45 min_in_list([[_,D0],[P1,D1]|T],(P,D)) :-
46     D0 > D1,
47     min_in_list([[P1,D1]|T],(P,D)).
```

Figure 6: *Prolog*-code for finding the shortest path from one cabin to another. The *"plan"* function is the same as in figure 4.

```
Distance = 10,
Path = [c1, c2]
```

```
?- bestplan(c1, c2, Path, Distance)
```

Figure 7: Result when running code in figure 6 with ”c1” to ”c2”.