NOEH, Noah-Vincenz (nn4718)

Imperial College London

Department of Computing
Academic Year **2018-2019**
Page created Thu Nov 1 19:15:32 GMT 2018

CATE

531 tk106 2
a5 nn4718 v1

0011800147 65

Electronic submission

Thu - 01 Nov 2018 18:41:58

nn4718

Exercise Information

Module:	531 Prolog	**Issued:**	Mon - 22 Oct 2018
Exercise:	2 (LAB)	**Due:**	Thu - 01 Nov 2018
Title:	Crossings	**Assessment:**	Individual
FAO:	Kimber, Timothy (tk106)	**Submission:**	Electronic

Student Declaration - Version 1

- I declare that this final submitted version is my unaided work.

Signed: (electronic signature) Date: 2018-11-01 18:41:46

For Markers only: (circle appropriate grade)

NOEH, Noah-Vincenz (nn4718)	01562775	a5	2018-11-01 18:41:46	**A*** **A** **B** **C** **D** **E** **F**

```
Prolog Crossings: Summary for nn4718 of a5
------------------------------------------

  Full Tests:
    Task 1 safe/1:      6 / 6
    Task 2 goal/1:      6 / 6
    Task 3 equiv/2:     7 / 7
    Task 4 visited/2:   3 / 3
    Task 5 choose/2:    4 / 4
    Task 6 journey/2:   4 / 4
    Task 7 succeeds/1:  1 / 1
    Task 8 fee/3:       2 / 2
    Task 9 cost/2:      1 / 1

```

```
  1: %% File: crossings.pl
  2: %% Name: Noah-Vincenz Noeh
  3: %% Date: 01/11/2018
  4: %%
  5: %% This program is a solution to Prolog 531 Assessed Exercise 2 'Crossings'
  6: %% The exercise is a version of the classic Farmer-Wolf-Goat-Cabbage Puzzle
  7:
  8: %% Step 1 safe(+Bank)
  9: % a bank is safe if 1) wolf and goat are together and there is no farmer or 2) goa
t and cabbage are together and there is no farmer
 10: safe(Bank)  :-
 11:   \+ (member(w, Bank), member(g, Bank), \+ member(f, Bank)),
 12:   \+ (member(g, Bank), member(c, Bank), \+ member(f, Bank)).
 13:
 14:
 15: %% Step 2 goal(+State)
 16: goal([]-SouthBank)  :-
 17:   length(SouthBank, 5),
 18:   member(g, SouthBank),
 19:   member(c, SouthBank),
 20:   member(w, SouthBank),
 21:   member(f, SouthBank),
 22:   member(b, SouthBank).
 23:
 24:
 25: %% Step 3 equiv(+State1, +State2)
 26: % this will match if all items are the same f.ex. all are empty
 27: equiv(A-A, A-A).
 28:
 29: % A and C must be the same length and B and D must be the same length
 30: equiv(A-B, C-D) :-
 31:   length(A, X),
 32:   length(C, X),
 33:   length(B, Y),
 34:   length(D, Y),
 35:   equiv_states(A-B, C-D).
 36:
 37: % if both A and B are non-empty
 38: equiv_states([HeadA|TailA]-[HeadB|TailB], C-D)  :-
 39:   member(HeadA, C),
 40:   member(HeadB, D),
 41:   equiv_states(TailA-TailB, C-D).
 42:
 43: % if A is empty
 44: equiv_states([]-[HeadB|TailB], C-D) :-
 45:   member(HeadB, D),
 46:   equiv_states([]-TailB, C-D).
 47:
 48: % if B is empty
 49: equiv_states([HeadA|TailA]-[], C-D) :-
 50:   member(HeadA, C),
 51:   equiv_states(TailA-[], C-D).
 52:
 53: % if A and B are both empty - note: this is only after items have been removed fro
m A and B
 54: equiv_states([]-[], _-_).
 55:
 56:
 57: %% Step 4 visited(+State, +Sequence)
 58: % if the state is equivalent to the head of the sequence X then succeed
 59: visited(State, [X|_]) :-
 60:   equiv(State, X).
 61:
 62: % else check if state is equivalent
 63: visited(State, [_|Tail])  :-
 64:   visited(State, Tail).
 65:
 66: % if the sequence is empty then we ca
 67: visited(_, [])  :-
 68:   fail.
 69:
 70:
 71: %% Step 5 choose(-Items, +Bank)
 72: % can only choose items if 'f' is a member of the bank
 73: choose(Items, Bank):-
 74:   member(f, Bank),
 75:   remove_list(Bank, [f], NewList),
 76:   choose_items(Items, [f], NewList, NewList),
 77:   remove_list(Bank, Items, NewerList),
ry single case
 78:   safe(NewerList).
 79:
 80: % we want to find any combination of another item together with 'f' that keeps the
 leftover items in the bank safe (possibly just 'f')
 81: choose_items(Items, Acc, [X|Tail], Bank) :-
 82:   length(Acc, N),
 83:   N < 2,
 84:   remove_list(Bank, [X|Acc], NewList),
 85:   safe(NewList),
 86:   choose_items(Items, [X|Acc], Tail, Bank).
 87:
 88: % This case is used when we already have two elements
o keep going through leftovers in bank but we want to output
 89: choose_items(Items, Acc, [_|_], Bank) :-
 90:   length(Acc, N),
 91:   N = 2,!,
 92:   choose_items(Items, Acc, [], Bank).
 93:
 94: % if any of the parts of the if clauses above fail then this is called - skips hea
d and checks next element
 95: choose_items(Items, Acc, [_|Tail], Bank) :-
 96:   choose_items(Items, Acc, Tail, Bank).
 97:
 98: choose_items(Acc, Acc, [], _).
 99:
100: % remove_list(ListA, ListB, ListC) removes all items in ListB from ListA and the o
utput is ListC
101: remove_list([], _, []).
102:
103: remove_list([X|Tail], ListB, ListOutput)  :-
104:   member(X, ListB), !,
105:   remove_list(Tail, ListB, ListOutput).
106:
107: remove_list([X|Tail], ListB, [X|ListOutput])  :-
108:   remove_list(Tail, ListB, ListOutput).
109:
110:
111: %% Step 6 journey(+State1, -State2)
112: % if f is a member of A then find items from choose append them to B in order to g
et D and remove them from A in order to get C - if any of these fail then fail
113: journey(A-B, C-D) :-
114:   member(f,A),!,
115:   choose(Items, A),
116:   append(B, Items, D),
117:   remove_list(A, Items, C).
118:
119: journey(A-B, C-D) :-
120:   member(f,B),
121:   choose(Items, B),
122:   append(A, Items, C),
123:   remove_list(B, Items, D).
124:
125:
126: %% Step 7 succeeds(-Sequence)
```

5/5

10/10

6/10

3/5

11/15

9/10

This is completely pointless.
You do not need to provide any clause
For this case.

Very confusing. You call this with Acc=[f] and you want max 2 items so why would you call it again?

Not needed. Just return items from previous clause

If you checked banks against each other you do not need all these cases

This is not sufficient.
[g,g,w]-[ ] is equiv to [g,w,w]-[ ] by this definition

Add a cut to prevent checking the whole sequence on backtracking.

Not "else" unless you use cut or conditional

Not needed. choose cannot succeed if f is not present.

```
  127: succeeds(Sequence) :-
  128:   extend([ [f,w,g,c,b]-[] ], ReversedSequence),
  129:   reverse(ReversedSequence, Sequence).  % the sequence was reversed by appending t
o the front of the sequence, so we need to reverse it in order get the sequence from star
t to end order
  130:
  131: % if the last state in the sequence (head) is a goal state then return
  132: extend([X|StatesVisitedSequence], [X|StatesVisitedSequence]) :-
  133:   goal(X).
  134:
  135: extend(StatesVisitedSequence, FinalSequence) :-
  136:   StatesVisitedSequence = [X|_],
  137:   journey(X, SomeNextState),
  138:   \+ visited(SomeNextState, StatesVisitedSequence),
  139:   extend([SomeNextState|StatesVisitedSequence], FinalSequence).
  140:
  141: % reverses the sequence as adding elements to the head of the sequence keeps the l
ast element at the front
  142: reverse(Sequence, NewSequence) :-
  143:   reverse_sequence(Sequence, [], NewSequence).
  144:
  145: reverse_sequence([], NewSequence, NewSequence).
  146:
  147: reverse_sequence([X|Tail], Acc, NewSequence) :-
  148:   reverse_sequence(Tail, [X|Acc], NewSequence).
  149:
  150:
  151: %% Step 8 fee(+State1, +State2, -Fee)
  152: % checks the difference of elements in the banks of the states to check whether th
e farmer travelled alone (cost 1) or with another item (cost 2)
  153: fee(A-B, C-D, Fee) :-
  154:   length(A, LengthA),
  155:   length(B, LengthB),
  156:   length(C, LengthC),
  157:   length(D, LengthD),
  158:   calc_fee(LengthA, LengthB, LengthC, LengthD, Fee).
  159:
  160: calc_fee(LengthA, _, LengthC, _, Fee) :-
  161:   LengthA is LengthC+1,!,
  162:   fees(Fee, _).
  163:
  164: calc_fee(_, LengthB, _, LengthD, Fee) :-
  165:   LengthB is LengthD+1,!,
  166:   fees(Fee, _).
  167:
  168: calc_fee(LengthA, _, LengthC, _, Fee) :-
  169:   LengthC is LengthA+1,!,
  170:   fees(Fee, _).
  171:
  172: calc_fee(_, LengthB, _, LengthD, Fee) :-
  173:   LengthD is LengthB+1,!,
  174:   fees(Fee, _).
  175:
  176: calc_fee(_, _, _, _, Fee) :-
  177:   fees(_, Fee).
  178:
  179: fees(1, 2).
  180:
  181:
  182: %% Step 9 cost(-Sequence, -Cost)
  183: cost(Sequence, Cost) :-
  184:   succeeds(Sequence),
  185:   calculate_cost(Sequence, 0, Cost).
  186:
  187: % uses an accumulator to add a fee for each trip between states
  188: calculate_cost([X|Tail], Acc, Cost) :-
  189:   Tail = [Y|_],!,
  190:   fee(X,Y,Fee),
  191:   NewAcc is Acc + Fee,
  192:   calculate_cost(Tail, NewAcc, Cost).
  193:
  194: calculate_cost(_, Cost, Cost).
```

9/10

Need a cut to avoid using next clause as well. All clauses that accept the inputs will be used on backtracking - no "else"

There is an abs operator you could use here

68/80. Well done. For recursive programs make sure that when the base case applies, the recursive case does not apply as well.

5/5

10/10

```
    1: % compiling /root/labcat/labcat/engines/lib/prolog/automarker.pl...
    2: %  loading /usr/lib/sicstus4.3.5/bin/sp-4.3.5/sicstus-4.3.5/library/timeout.po...
    3: %  module timeout imported into user
    4: %   loading /usr/lib/sicstus4.3.5/bin/sp-4.3.5/sicstus-4.3.5/library/types.po...
    5: %   module types imported into timeout
    6: %   loaded /usr/lib/sicstus4.3.5/bin/sp-4.3.5/sicstus-4.3.5/library/types.po in mo
dule types, 0 msec 4112 bytes
    7: %   loading foreign resource /usr/lib/sicstus4.3.5/bin/sp-4.3.5/sicstus-4.3.5/libr
ary/x86_64-linux-glibc2.17/timeout.so in module timeout
    8: %  loaded /usr/lib/sicstus4.3.5/bin/sp-4.3.5/sicstus-4.3.5/library/timeout.po in m
odule timeout, 10 msec 52576 bytes
    9: % compiled /root/labcat/labcat/engines/lib/prolog/automarker.pl in module user, 22
0 msec 1055536 bytes
   10: SICStus 4.3.5 (x86_64-linux-glibc2.17): Tue Dec  6 10:41:06 PST 2016
   11: Licensed to SP4.3doc.ic.ac.uk
   12: % compiling /tmp/d20181101-35-vi9r9f/src/crossings_auto_patch.pl...
   13: % compiled /tmp/d20181101-35-vi9r9f/src/crossings_auto_patch.pl in module user, 0
msec 16048 bytes
   14: % compiling /tmp/d20181101-35-vi9r9f/src/solution.pl...
   15: %  loading /usr/lib/sicstus4.3.5/bin/sp-4.3.5/sicstus-4.3.5/library/lists.po...
   16: %  module lists imported into model
   17: %   module types imported into lists
   18: %  loaded /usr/lib/sicstus4.3.5/bin/sp-4.3.5/sicstus-4.3.5/library/lists.po in mod
ule lists, 10 msec 127040 bytes
   19: % compiled /tmp/d20181101-35-vi9r9f/src/solution.pl in module model, 10 msec 13990
4 bytes
   20: % compiling /tmp/d20181101-35-vi9r9f/src/crossings.pl...
   21: % compiled /tmp/d20181101-35-vi9r9f/src/crossings.pl in module submitted, 10 msec
14096 bytes
   22: yes
   23: yes
   24: yes
   25: yes
   26: yes
   27: yes
   28: yes
   29: yes
   30: yes
   31: yes
   32: yes
   33: yes
   34: yes
   35: yes
   36: yes
   37: yes
   38: yes
   39: yes
   40: yes
   41: yes
   42: yes
   43: yes
   44: yes
   45: yes
   46: yes
   47: yes
   48: yes
   49: yes
   50: yes
   51: yes
   52: yes
   53: yes
   54: yes
   55: yes
   56: yes
   57: yes
   58: yes
   59: yes
   60: yes
   61: yes
   62: yes
   63: yes
   64: yes
   65: yes
   66: yes
   67: yes
   68: yes
   69: yes
   70: yes
```

```
================================================================================
        531 Prolog: Exercise 2 (Crossings)
        Submission: nn4718
================================================================================

================================================================================
Task 1 safe/1

---------- Test 1 :: safe:: full bank safe ------------------------------------

| ? safe([b,c,f,g,w]).
yes        %% correct

---------- Test 2 :: safe:: empty bank safe -----------------------------------

| ? safe([]).
yes        %% correct

---------- Test 3 :: safe:: farmer only safe ----------------------------------

| ? safe([f]).
yes        %% correct

---------- Test 4 :: safe:: all except farmer unsafe --------------------------

| ? safe([b,c,g,w]).
no         %% correct

---------- Test 5 :: safe:: goat alone safe -----------------------------------

| ? safe([g]).
yes        %% correct

---------- Test 6 :: safe:: bag alone safe ------------------------------------

| ? safe([b]).
yes        %% correct


================================================================================
Task 1 safe/1
TESTS PASSED:  6 / 6
================================================================================

================================================================================
Task 2 goal/1

---------- Test 1 :: goal:: legal goal state ----------------------------------

| ? goal([]-[b,c,f,g,w]).
yes        %% correct

---------- Test 2 :: goal:: single bank not a state ---------------------------

| ? goal([b,c,f,g,w]).
no         %% correct

---------- Test 3 :: goal:: extra element other bank --------------------------

| ? goal([b]-[b,c,f,g,w]).
no         %% correct

---------- Test 4 :: goal:: legal non-goal state ------------------------------

| ? goal([b,c,f]-[g,w]).
no         %% correct

---------- Test 5 :: goal:: wolf is missing -----------------------------------

| ? goal([]-[b,c,f,g]).
no         %% correct

---------- Test 6 :: goal:: extra element same bank ---------------------------

| ? goal([]-[b,c,f,g,w,a]).
no         %% correct


================================================================================
Task 2 goal/1
TESTS PASSED:  6 / 6
================================================================================

================================================================================
Task 3 equiv/2

---------- Test 1 :: equiv:: single bank not a state --------------------------

| ? equiv([b,c,f,g,w],[b,c,f,g,w]).
no         %% correct

---------- Test 2 :: equiv:: single empty bank not a state --------------------

| ? equiv([],[]).
no         %% correct

---------- Test 3 :: equiv:: legal reordered goal states ----------------------

| ? equiv([]-[b,c,f,g,w],[]-[w,g,f,c,b]).
yes        %% correct

---------- Test 4 :: equiv:: legal reordered state ----------------------------

| ? equiv([b,c]-[f,g,w],[c,b]-[w,g,f]).
yes        %% correct

---------- Test 5 :: equiv:: North equiv South not ----------------------------

| ? equiv([g]-[b,c,f],[g]-[w,f,c,b]).
no         %% correct

---------- Test 6 :: equiv:: South equiv North not ----------------------------

| ? equiv([g,g]-[b,c,f,w],[g]-[w,f,c,b]).
no         %% correct

---------- Test 7 :: equiv:: correct but distinct states ----------------------

| ? equiv([b,c]-[f,g,w],[c,g]-[w,b,f]).
no         %% correct


================================================================================
Task 3 equiv/2
TESTS PASSED:  7 / 7
================================================================================

================================================================================
Task 4 visited/2

---------- Test 1 :: visited:: non states -------------------------------------

| ? visited([b,c,f,g,w],[b,c,f,g,w]).
no         %% correct
```

```
135:
136: ---------- Test 2 :: visited:: sequence with equiv state ----------------------
137:
138: | ? visited([b,c]-[f,g,w],[[]-[b,c,f,g,w],[c,f]-[b,g,w],[c]-[b,f,g,w],[c,b]-[f,g,w
],[c,f]-[b,g,w]]).
139: yes        %% correct
140:
141: ---------- Test 3 :: visited:: no equiv present -------------------------------
142:
143: | ? visited([g,c]-[f,b,w],[[]-[b,c,f,g,w],[c,f]-[b,g,w],[c]-[b,f,g,w],[c,b]-[f,g,w
],[c,f]-[b,g,w]]).
144: no         %% correct
145:
146:
147: ================================================================================
148: Task 4 visited/2
149: TESTS PASSED:  3 / 3
150: ================================================================================
151:
152: ================================================================================
153: Task 5 choose/2
154:
155: ---------- Test 1 :: choose:: all 4 possibles are safe ------------------------
156:
157: | ? find I: choose(I,[b,c,f,w]).
158:   I = [b,f] ;
159:   I = [c,f] ;
160:   I = [w,f] ;
161:   I = [f] ;
162: No more solutions    (All correct)
163: No missing solutions
164:
165: ---------- Test 2 :: choose:: only 1 possible safe ----------------------------
166:
167: | ? find I: choose(I,[b,c,f,g,w]).
168:   I = [g,f] ;
169: No more solutions    (All correct)
170: No missing solutions
171:
172: ---------- Test 3 :: choose:: the only possible is safe -----------------------
173:
174: | ? find I: choose(I,[f]).
175:   I = [f] ;
176: No more solutions    (All correct)
177: No missing solutions
178:
179: ---------- Test 4 :: choose:: farmer not present ------------------------------
180:
181: | ? find I: choose(I,[w,c,b]).
182: No solution         %% correct
183:
184:
185: ================================================================================
186: Task 5 choose/2
187: TESTS PASSED:  4 / 4
188: ================================================================================
189:
190: ================================================================================
191: Task 6 journey/2
192:
193: ---------- Test 1 :: journey:: 4 possible south to north ----------------------
194:
195: | ? find S: journey([g]-[b,c,f,w],S).
196:   S = [g,b,f]-[c,w] ;
197:   S = [g,c,f]-[b,w] ;
198:   S = [g,w,f]-[b,c] ;
199:   S = [g,f]-[b,c,w] ;
200: No more solutions    (All correct)
201: No missing solutions
202:
203: ---------- Test 2 :: journey:: 2 possible north to south ----------------------
204:
205: | ? find S: journey([f,g]-[b,c,w],S).
206:   S = []-[b,c,w,g,f] ;
207:   S = [g]-[b,c,w,f] ;
208: No more solutions    (All correct)
209: No missing solutions
210:
211: ---------- Test 3 :: journey:: 1 possible south to north ----------------------
212:
213: | ? find S: journey([b,c,g,w]-[f],S).
214:   S = [b,c,g,w,f]-[] ;
215: No more solutions    (All correct)
216: No missing solutions
217:
218: ---------- Test 4 :: journey:: 1 possible north to south ----------------------
219:
220: | ? find S: journey([b,c,g,w,f]-[],S).
221:   S = [b,c,w]-[g,f] ;
222: No more solutions    (All correct)
223: No missing solutions
224:
225:
226: ================================================================================
227: Task 6 journey/2
228: TESTS PASSED:  4 / 4
229: ================================================================================
230:
231: ================================================================================
232: Task 7 succeeds/1
233:
234: ---------- Test 1 :: succeeds:: 4 distinct possibles --------------------------
235:
236: | ? find Seq: succeeds(Seq).
237:   Seq = [[f,w,g,c,b]-[],[w,c,b]-[g,f],[w,c,b,f]-[g],[c,b]-[g,w,f],[c,b,g,f]-[w],[b
,g]-[w,c,f],[b,g,f]-[w,c],[g]-[w,c,b,f],[g,f]-[w,c,b],[]-[w,c,b,g,f]] ;
238:   Seq = [[f,w,g,c,b]-[],[w,c,b]-[g,f],[w,c,b,f]-[g],[w,b]-[g,c,f],[w,b,g,f]-[c],[b
,g]-[c,w,f],[b,g,f]-[c,w],[g]-[c,w,b,f],[g,f]-[c,w,b],[]-[c,w,b,g,f]] ;
239:   Seq = [[f,w,g,c,b]-[],[w,c,b]-[g,f],[w,c,b,f]-[g],[w,c]-[g,b,f],[w,c,f]-[g,b],[c
]-[g,b,w,f],[c,g,f]-[b,w],[g]-[b,w,c,f],[g,f]-[b,w,c],[]-[b,w,c,g,f]] ;
240:   Seq = [[f,w,g,c,b]-[],[w,c,b]-[g,f],[w,c,b,f]-[g],[w,c]-[g,b,f],[w,c,f]-[g,b],[w
]-[g,b,c,f],[w,g,f]-[b,c],[g]-[b,c,w,f],[g,f]-[b,c,w],[]-[b,c,w,g,f]] ;
241: No more solutions    (All correct)
242: No missing solutions
243:
244:
245: ================================================================================
246: Task 7 succeeds/1
247: TESTS PASSED:  1 / 1
248: ================================================================================
249:
250: ================================================================================
251: Task 8 fee/3
252:
253:
254:
255: ************************************
256: The remaining tests will be carried out using:
257:     fees( 5, 7 ).
258: ************************************
259:
260: ---------- Test 1 :: fee:: farmer travels alone -------------------------------
261:
262: | ? find Fee: fee([f,g]-[b,c,w],[g]-[f,b,c,w],Fee).
```

```
  Fee = 5 ;
No more solutions    (All correct)
No missing solutions

---------- Test 2 :: fee:: farmer plus item ----------------------------------

| ? find Fee: fee([f,g]-[b,c,w],[]-[f,g,b,c,w],Fee).
  Fee = 7 ;
No more solutions    (All correct)
No missing solutions


================================================================================
Task 8 fee/3
TESTS PASSED:  2 / 2
================================================================================

================================================================================
Task 9 cost/2

---------- Test 1 :: cost:: four sequences should cost same --------------------

| ? find Cost: cost(_999,Cost).
  Cost = 57 ;
  Cost = 57 ;
  Cost = 57 ;
  Cost = 57 ;
No more solutions    (All correct)
No missing solutions


================================================================================
Task 9 cost/2
TESTS PASSED:  1 / 1
================================================================================


============================== SUMMARY (nn4718) ==============================

TESTS PASSED:  34 / 34
================================================================================
```