10/24/14 11:20:38 /home-reseau/helomari/4INFO/4INFO/Prolog/tp6/tp6_turing_etud.ecl

```
 1
 2  %%%%%%%%%% First part
 3
 4  copy_prog(program(
 5                       start,
 6                       [stop],
 7                       [delta(start, ' ', ' ', right, stop),
 8                        delta(start, 1, ' ', right, s2),
 9                        delta(s2, 1, 1, right, s2),
10                        delta(s2, ' ', ' ', right, s3),
11                        delta(s3, 1, 1, right, s3),
12                        delta(s3, ' ', 1, left, s4),
13                        delta(s4, 1, 1, left, s4),
14                        delta(s4, ' ', ' ', left, s5),
15                        delta(s5, 1, 1, left, s5),
16                        delta(s5, ' ', 1, right, start)
17                       ]
18                   )
19           ).
20
21  initial_state(program(InitialState, _, _), InitialState).
22
23  final_states(program(_, FinalStates, _), FinalStates).
24
25  transitions(program(_, _, Deltas), Deltas).
26
27
28  %write to meta post format
29  %compile result with:
30  % mpost filename
31  % epstopdf filename.1
32  dump_to_mpost(Filename, Dump) :-
33          open(Filename, append, Stream),
34              write_header(Stream),
35          write_dump(0, Dump, Stream),
36          write_end(Stream),
37          close(Stream).
38
39  write_header(Stream) :-
40          write(Stream, 'prologues := 1;\n'),
41          write(Stream, 'input turing;\n'),
42          write(Stream, 'beginfig(1)\n').
43
44  write_end(Stream) :-
45          write(Stream, 'endfig;\n'),
46          write(Stream, 'end').
47
48  write_dump(_, [], _).
49  write_dump(Y, [(State, Tape) | Tapes], Stream) :-
50          write(Stream, 'tape(0, '),
51          write(Stream, Y),
52          write(Stream, 'cm, 1cm, \"'),
53          write(Stream, State),
54          write(Stream, '\", '),
55          write_tape(Tape, Stream),
56          write(Stream, ');\n'),
57          Y1 is Y - 2,
58          write_dump(Y1, Tapes, Stream).
59
60  write_tape(tape(Left, Right), Stream) :-
61          length(Left, N),
62          write(Stream, '\"'),
63          append(Left, Right, L),
64          (param(Stream), foreach(X, L) do
65              write(Stream, X)
66          ),
67          write(Stream, '\", '),
68          write(Stream, N),
69          write('\n').
70  /*
71  ["~/4INFO/Prolog/tp6/tp6_turing_etud.ecl"].
72  */
73  /*
74    Question 1.1 : next permet de faire une transition d'un état à un autre
75  */
76  next(Program, State0, Symbole0, Symbole1, Dir, State1):-transitions(Program, Deltas),compare(Deltas,
    State0, Symbole0, Symbole1, Dir, State1).
77
78  compare([delta(State0,Symbole0,Symbole1,Dir,State1)|Deltas], State0, Symbole0, Symbole1, Dir, State1).
79  compare([delta(St0,S0,S1,Direction,St1)|Deltas], State0, Symbole0, Symbole1, Dir, State1):- \==
    (St0,State0),compare(Deltas, State0, Symbole0, Symbole1, Dir, State1).
80  compare([delta(St0,S0,S1,Direction,St1)|Deltas], State0, Symbole0, Symbole1, Dir, State1):- \==
    (S0,Symbole0),compare(Deltas, State0, Symbole0, Symbole1, Dir, State1).
81
```

```
82  /*
83    Question 1.2 : update_tape : mettre à jour la bande de la machine
84  */
85  update_tape(tape(Left,[_]), Symb, right, tape(L, [' '])):-append(Left,[Symb],L).
86  update_tape(tape(Left,[_|Right]), Symb, right, tape(L, Right)):-append(Left,[Symb],L).
87  update_tape(tape(Left,[_|Right]), Symb, left, tape(L, R)):- insert(Symb,Right,R1), deplacer(Left,R1,L,R).
88
89  insert(Symb,Right,[Symb|Right]).
90
91  deplacer([A],R1,[],[A|R1]).
92  deplacer([A|Left],R1,[A|L],R):- deplacer(Left,R1,L,R).
93
94  /*
95    Question 1.3 : Executer le programme avec input comme symbole d'entrée
96  */
97  run_turing_machine(Program, [Symb|Input], Output, FinalState):-initial_state(Program,
    InitialState),next(Program,InitialState,Symb,Symb1,Dir,State1),update_tape(tape([' '],[Symb|Input]),
    Symb1, Dir, tape(Left,Right)),run_turing_machine_rec(Program,Left,Right,State1,Output,FinalState),!.
98
99  run_turing_machine_rec(Program, Left, Right, State0, Output, State0):-final_states(Program,
    FinalStates),member(State0,FinalStates),append(Left,Right,Output).
100
101 run_turing_machine_rec(Program, Left, [Symb|Input], State0, Output, FinalState):-
    next(Program,State0,Symb,Symb1,Dir,State1), update_tape(tape(Left,
    [Symb|Input]),Symb1,Dir,tape(Left1,Right1)),run_turing_machine_rec(Program,Left1,Right1,State1,Output,Fina
    lState).
102
103 /*
104   Question 1.4 : produire une liste représentant les différentes étapes de l'exécution de la machine de
    turing
105 */
106 run_turing_machine(Program, [Symb|Input], Output, FinalState, Dump):-initial_state(Program,
    InitialState),next(Program,InitialState,Symb,Symb1,Dir,State1),update_tape(tape([' '],[Symb|Input]),
    Symb1, Dir, tape(Left,Right)),
107 append_strings("turing","0",Name), dump_to_mpost(Name,[(State0,tape([' '],[Symb|Input]))]),
108 run_turing_machine_rec(Program,Left,Right,State1,Output,FinalState, Dump, 1),!.
109
110 run_turing_machine_rec(Program, Left, Right, State0, Output, State0, Dump, Nb):- final_states(Program,
    FinalStates),member(State0,FinalStates),
111 number_string(Nb,Nb_str), append_strings("turing",Nb_str,Name), dump_to_mpost(Name,
    [(State0,tape(Left,Right))]),
112 append(Left,Right,Output).
113
114
115 run_turing_machine_rec(Program, Left, [Symb|Input], State0, Output, FinalState, Dump, Nb):-
116 next(Program,State0,Symb,Symb1,Dir,State1), update_tape(tape(Left,
    [Symb|Input]),Symb1,Dir,tape(Left1,Right1)), number_string(Nb,Nb_str),
    append_strings("turing",Nb_str,Name), dump_to_mpost(Name,[(State0,tape(Left,[Symb|Input]))]), Nb1 is Nb +
    1, run_turing_machine_rec(Program,Left1,Right1,State1,Output,FinalState,Dump, Nb1).
117
118
119
120 /*
121 %Question 1.1
122   copy_prog(Program),next(Program,start,1,NewS,Dir,NextSt).
123     Program = program(start, [stop], [delta(start, ' ', ' ', right, stop), delta(start, 1, ' ', right,
    s2), delta(s2, 1, 1, right, s2), delta(s2, ' ', ' ', right, s3), delta(s3, 1, 1, right, s3), delta(s3, '
    ', 1, left, s4), delta(s4, 1, 1, left, s4), delta(s4, ' ', ' ', left, s5), delta(s5, 1, 1, left, s5),
    delta(s5, ' ', 1, right, start)])
124     NewS = ' '
125     Dir = right
126     NextSt = s2
127     Yes (0.00s cpu, solution 1, maybe more)
128
129 %Question 1.2
130   update_tape(tape([' '],[1,' ']),' ',right,UpdatedTape).
131     UpdatedTape = tape([' ', ' '], [' '])
132     Yes (0.00s cpu)
133   update_tape(tape([' ',' ',' '],[' ']),1,left,UpdatedTape).
134     UpdatedTape = tape([' ', ' '], [' ', 1])
135     Yes (0.00s cpu)
136
137 %Question 1.3
138   copy_prog(P),run_turing_machine(P,[1],Output,FinalState).
139     P = program(start, [stop], [delta(start, ' ', ' ', right, stop), delta(start, 1, ' ', right, s2),
    delta(s2, 1, 1, right, s2), delta(s2, ' ', ' ', right, s3), delta(s3, 1, 1, right, s3), delta(s3, ' ', 1,
    left, s4), delta(s4, 1, 1, left, s4), delta(s4, ' ', ' ', left, s5), delta(s5, 1, 1, left, s5), delta(s5,
    ' ', 1, right, start)])
140     Output = [' ', 1, ' ', 1]
141     FinalState = stop
142     Yes (0.00s cpu)
143
144 %Question 1.4
145 */
146 %%%%%%%%%% Optional part
147
148 %make_pairs(+, -): 'a list * ('a * 'a) list
```

```
149  make_pairs([], _, []).
150  make_pairs([X | L], L2, Res) :-
151          make_pairs_aux(X, L2, Pairs),
152          make_pairs(L, L2, RemainingPairs),
153          append(Pairs, RemainingPairs, Res).
154
155  %make_pairs_aux(+, +, -): 'a * 'a list * ('a * 'a) list
156  make_pairs_aux(_, [], []).
157  make_pairs_aux(X, [Y | Ys], [(X, Y) | Zs]) :-
158          make_pairs_aux(X, Ys, Zs).
159
160  complete(S1, Sym, Symbols, Directions, States, Res) :-
161          member(Sym1, Symbols),
162          member(Dir, Directions),
163          member(S2, States),
164          Res = delta(S1, Sym, Sym1, Dir, S2).
165
166  complete_list([], _, _, _, []).
167  complete_list([(S, Sym) | Pairs], Symbols, Directions, States, [Delta | Deltas]) :-
168          complete(S, Sym, Symbols, Directions, States, Delta),
169          complete_list(Pairs, Symbols, Directions, States, Deltas).
```