Durbala Cristian 333AC

Tema 2 – Maze

Automatul este realizat printr-un algoritm de tipul Wall Follower. Acesta urmareste peretele din dreapta pentru a ajunge la iesirea labirintului.

Ma folosesc de 8 stari pentru a verifica daca intalnesc un perete sau o celula libera.

Fiecare verificare realizata intr-o anumita directie consta in parcurgerea (by default) a doua stari:

- **Starea de tipul verif_*,** unde * = {jos, dreapta, sus, stanga}

  Aici este realizata incrementarea/decrementarea randului sau a coloanei, in functie de directia in care vreau sa ma misc. In fiecare stare de tip verif_* se activeaza semnalul maze_oe, deoarece urmeaza sa ma folosesc de maze_in la ciclul urmator.

- **Starea de tipul verif_* + 1**

  Aici verific daca celula in care am ajuns este libera, sau daca am intalnit un perete.
  In cazul in care am un perete (maze_in == 1), trec la verificarea urmatoarei celule, in sens trigonometric (pentru a ramane langa peretele din dreapta).
  Daca celula in care am ajuns nu contine un perete (maze_in == 0), activez semnalul maze_we pentru a marca trecerea si merg la urmatoarea verificare. Tot in acest caz verific si daca am ajuns pe marginea labirintului (conditie iesire din labirint), caz in care trec la starea “iesire”.

Celelalte 2 stari folosite in cadrul automatului sunt:

- **Starea init**

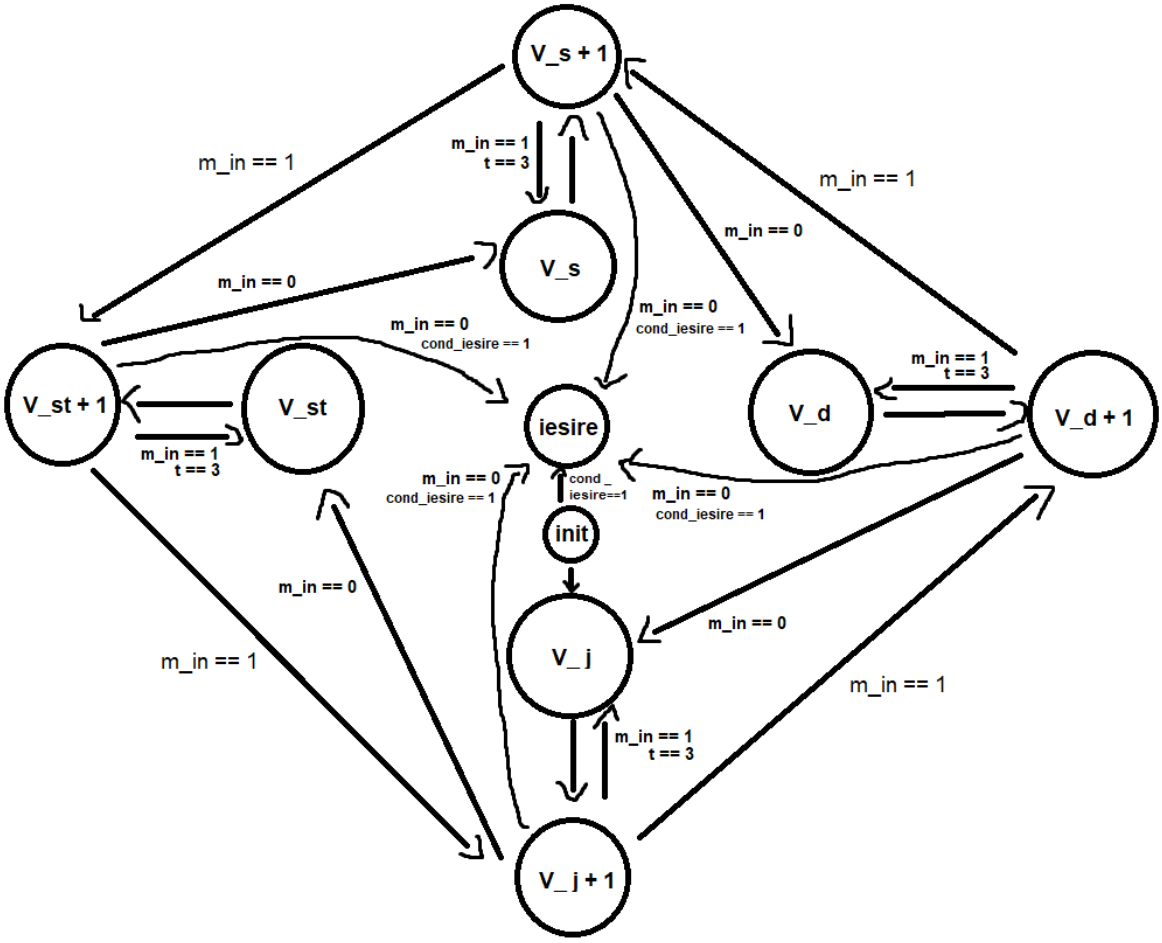
  Aceasta stare este parcursa o singura data. Aici se egaleaza row si col cu intrarile starting_row si starting_col. Activez semnalul maze_we pentru marcarea trecerii.

  Am adaugat conditia de iesire pentru cazul neintalnit, chiar ridicol, in care plec exact din iesirea labirintului.

- **Starea iesire**

  Desi in aceasta stare nu se intampla nimic, aceasta este necesara pentru a asigura ca automatul nu incepe sa parcurga din nou celelalte stari chiar si dupa activarea semnalului done.

Toate aceste stari sunt conectate conform reprezentarii in urmatoarea schema:



Modalitati relevante scadere numar de cicli

1. Eliminarea variabilei de directie

Initial, modulul meu continea o variabila numita “directie”, in care stocam ultima directie in care ma deplasam in labirint, la momentul gasirii unei celule libere(fara perete).

Aceasta era insotita de o stare numita verif_directie, prin care automatul meu trecea de fiecare data cand gasea o celula libera, pentru a determina in ce parte trebuie sa verifice intai (adica ce inseamna pentru el "dreapta" la momentul respectiv).

```
parameter jos = 0;
parameter dreapta = 1;
parameter sus = 2;
parameter stanga = 3;
```

```
`verif_directie: begin
   if (directie == jos) begin
      state_next = verif_stanga;
   end
   else if (directie == dreapta) begin
      state_next = verif_jos;
   end
   else if (directie == sus) begin
      state_next = verif_dreapta;
   end
   else if (directie == stanga) begin
      state_next = verif_sus;
   end
end
```

Eventual, am realizat ca aceasta stare nu este necesara, deoarece pot sa trimit manual automatul in starea necesara direct din verificari.

Exemplu:

Daca m_in == 0 in verif_sus + 1, state_next = verif_dreapta.

Numar cicli castigati: 1 la fiecare celula libera intalnita

2. Modificare row/col direct in verif_* + 1

Urmatorul mod de a scadea numarul de cicli necesar rezolvarii labirintului este prin asigurarea faptului ca automatul meu face intotdeauna ceva, indiferent de situatie. Mai exact, chiar daca nu pot modifica row sau col pentru pasul urmator in cazul in care gasesc o celula libera (din cauza lui maze_we = 1), acest lucru este permis in cazul in care intalnesc un perete. Din moment ce sar peste o stare de tip verif_*, tot aici trebuie sa activez maze_oe.

Exemplu:

In cazul in care am maze_in == 1 in verif_jos, dupa ce revin la pozitia initiala prin realizarea operatiei row = row – 1, pot sa fac si operatia col = col + 1 si sa trec direct la verif_dreapta + 1 in loc sa mai trec prin verif_dreapta.

Numar cicli castigati: 1 la fiecare perete intalnit

3. Adaugarea verificarii numarului de pereti intalniti de la ultima miscare

Pentru a scadea si mai mult numarul de cicli necesari, am adaugat variabila t, care imi verifica numarul de pereti intalniti de ultima data cand m-am miscat. Aceasta este initializata cu 0 si este incrementata de fiecare data cand intalnesc in starile de tip verif_* + 1 conditia maze_in == 1.

Cand t ajunge sa ia valoarea 3 (am intalnit al 3-lea perete la rand) ma pot afla in 2 situati: Ori sunt in punctul de plecare, ori am ajuns intr-o fundatura. In ambele cazuri, a 4-a celula ce urmeaza a fi verificata va fi mereu un spatiu liber, deoarece nu am cum sa fiu inconjurat de 4 pereti. In urma realizarii miscarii, t va fi reinitializat cu 0, indiferent daca ajunsese sau nu sa ia valoarea 3.

Desi nu ar fi necesara verificarea conditiei de iesire in mod normal (dintr-o fundatura ma pot intoarce doar pe unde am venit, campul respectiv este mai mult ca sigur confirmat la pasii anteriori ca nu este iesire), am adaugat-o pentru situatiile (neintalnite in teste, dar posibile) in care punctul meu de plecare este chiar la o celula distanta de iesirea labirintului (ex. incep la [15,1] si iesirea este [15, 0]).

Exemplu:

Sunt pe pozitia [50, 50] dupa ce ultima data m-am miscat in jos. Automatul verifica pozitiile [50, 49], [51, 50] si [50, 51] (stanga, jos, dreapta) si intalneste pereti in toate 3 pozitiile. In urma intaliniri celui de-al 3-lea perete, acesta merge in sus, la pozitia [49, 50] (garantat libera) si pleaca direct in verif_dreapta fara a mai verifica daca se afla in perete sau nu.

Numar cicli castigati: 1 la fiecare fundatura, aproximativ 40-50 la fiecare test complex sau 0-2 la testele simple, 270 total

Note. Am fost putin ezitant la adaugarea acestui “artificiu” deoarece, desi este destul de logic, nu stiam daca se apropie sau nu destul de mult de lucrurile pe care nu aveam voie sa le facem (ex. cache-uire labirint).

Modalitati mai putin relevante scadere numar de cicli

1. Activarea iesirii done, de la momentul verificarii

La momentul intalnirii conditiei de ((row == 63) || (row == 0) || (col == 63) || (col == 0)), ar avea cel mai mult sens ca automatul meu sa mearga intr-o stare finala pentru egalarea valorii done cu 1. Totusi, daca activez done chiar din momentul verificarii, pot sa scap de un ciclu la fiecare test.

Numar cicli castigati: 1 la fiecare test, 10 total

2. Orientarea initiala in jos

Faptul ca stiu din cerinta ca pornesc inconjurat de 3 pereti inseamna ca aceasta modificare, din punct de vedere al functionarii in sine a automatului, nu face nimic. In cazul testelor oferite de checker-ul nostru, in schimb, pot obtine un avantaj extrem de mic la numarul total de cicli daca fac prima verificare dupa starea initiala in jos.

	jos	dreapta	sus	stanga
base	17	19	19	18
complex1	7290	7289	7288	7290
complex2	8896	8898	8898	8897
complex3	7344	7343	7345	7345
complex4	5763	5763	5762	5761
complex5	8739	8739	8738	8737
simple1	269	268	270	270
simple2	627	629	629	628
simple3	1269	1268	1267	1269
simple4	859	858	860	860
Total	41073	41074	41076	41075

Numar cicli castigati: 1-3 total