README Tema 2 AC – Maze

Marin Vanessa-Ramona 331AA

Pentru rezolvarea temei mi-am declarat:

- maze_width = 6 ca parametru pentru dimensiunea row, copy_row, col si copy_col, este egal cu 6 deoarece 63 scris pe biti are 6 biti
- state_width = 5, deoarece ‘h11 reprezentat pe biti are 5 biti
- direction_width = 3, deoarece 4 reprezentat pe biti are 4 biti
- margin = (verific daca am ajuns pe marginea labirintului),
  valoarea 0 - inca ma aflu in labirint, 1 - am ajuns pe marginea acestuia.

Pentru a hotara directia in care ma deplasez, mi-am luat direction care poate avea valorile:

➔ 1 - > merg in jos
➔ 2 -> merg in sus
➔ 3 -> merg in dreapta
➔ 4 -> merg in stanga

Pentru a-mi salva pozitia mi-am luat un copy_row si un copy_col.

Iar pentru trecerea dintr-o stare in alta, am: current_state si next_state

In total am 5 stari.

Pentru implementarea rezolvarii, m-am ajutat de Laboratorul 4 de AC de pe ocw, in care ne este descris modul de implementare a unui FSM, in cazul de fata o masina Moore:

➔ O sa avem o parte secventiala – modeleaza partea de stare
➔ Si inca o parte combinationala – modeleaza logica combinationala

In partea secventiala verific daca n-am ajuns la final, adica **done = 0**, atunci trec la urmatoarea stare pana ajung sa ies din labirint, adica **done = 1**.

In partea combinationala, tratez fiecare stare in parte, precum urmeaza:

**input_start:**

- Imi aleg o directie de deplasare, initiala, adica 1 (in jos)
- Imi setez pozitia initiala data
- Imi salvez o copie a acestei pozitii
- Marchez pozitia cu 2, pentru a sti ca am fost acolo
- Trec la urmatoarea stare

**check_direction_to_right:**

- Verific daca am ajuns pe marginea labirintului, daca da, margin devine 1, daca nu, margin ramane 0

- In continuare verific daca margin = 1, atunci trec direct in starea finala care ma anunta ca am iesit din labirint; Daca margin = 0, atunci verific in functie de directia de deplasare anterioara ce am in dreapta acesteia, salvand pozitia
- La sfarsitul aceste stari, voi permite citire pozitiei (row, col) si trec la urmatoarea stare

**check_wall_right:**

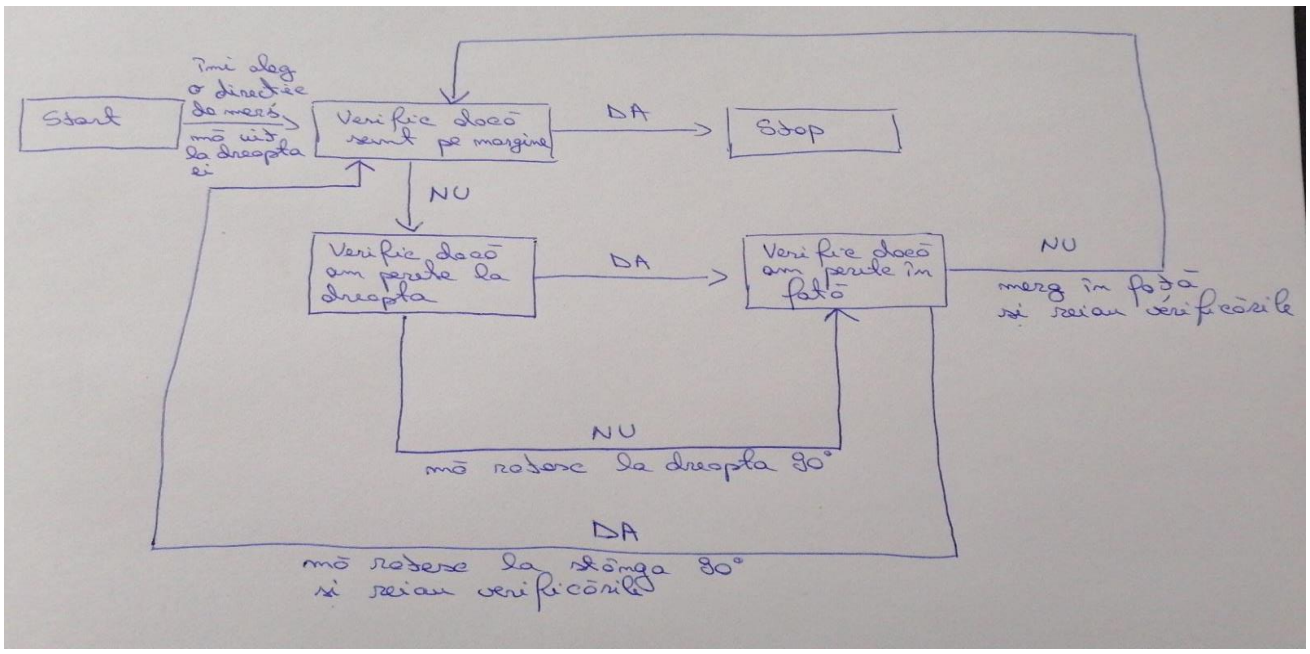
- In functie de directia de deplasare aleasa anterior, voi verifica daca am sau nu perete la dreapta, folosind **maze_in** astfel**:**
- **maze_in = 0 (**nu am perete in dreapta) imi voi pastra pozitia si ma voi roti 90 de grade la dreapta (robotului) si ma voi duce sa verific daca am perete in fata
- **maze_in = 1** (am perete la dreapta) ma voi duce sa verific in fata (in fata insemnand +1 o pozitie in directia aleasa), nu inainte de a-mi salva pozitia
- Permit citirea pozitiei si trec la urmatoarea stare

**check_front_wall:**

- Verific daca am perete in fata folosindu-ma de **maze_in**
- **maze_in = 0** (nu am perete in fata), ma mut o pozitie in fata, o marchez si reiau verificarile
- **maze_in = 1** (am perete in fata), in functie de directia anterioara de deplasare, ma rotesc 90 de grade la stanga (robotului), si salvez pozitia pe care ma aflu, apoi reiau verificarile
- In final reiau verificarile, in functie de directie, ma orientez la dreapta, verific daca am perete la dreapta, dupa verific daca am perete in fata. (acest lucru se face repetitiv pana ajung pe margine si ies din labirint)

**out:**

- Aceasta stare marcheaza faptul ca am ajuns sa ies din labirint, adica **done = 1**




| Directia de mers: |
|---|

**Ma uit la Robot de sus:**