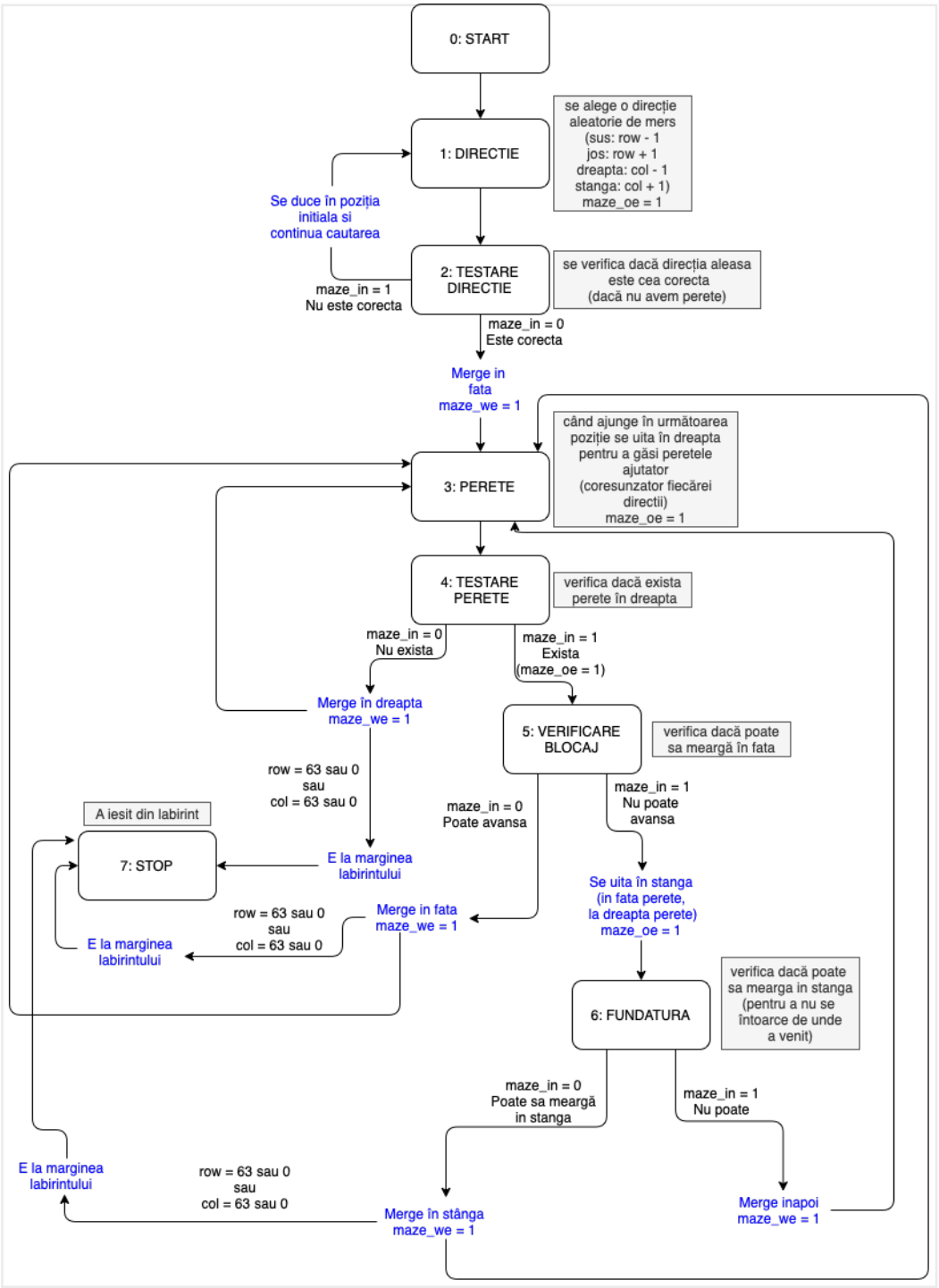
Mamut Nuray 331AB

Maze - Tema 2 AC

Schema functionare algoritm Wall Follower:



*CORECTIE SCHEMA: starea FUNDATURA -> cand merge inapoi nu se activeaza maze_we

Prezentare stari automat:

- **0: START - Default**
  - se initializeaza row si col cu valorile starting_row si starting_col
  - se pastreaza aceste valori in r_ant si c_ant (variabile folosite pentru pastrarea randului anterior si coloanei anterionare)
  - se activeaza scriere (maze_we = 1)
  - urmeaza sa ne ducem in starea **1** (next_state = `directie)

- **1: DIRECTIE**
  - exista 4 directii posibile de mers => 4 cazuri (fiecarei directii dir i se atribuie un caz)
    - 0: sus (row - 1)
    - 1: jos (row + 1)
    - 2: dreapta (col - 1)
    - 3: stanga (col + 1)
  - initial se va incerca mersul in sus (dir este implicit 0)
  - se activeaza citirea (maze_oe = 1), pentru a putea verifica daca directia aleasa este cea corecta (daca in fata nu este perete)
  - urmeaza starea **2** (next_state = `testarea_directiei)

- **2: TESTARE_DIRECTIE**
  - exista 2 posibilitati:
    1. directia aleasa este cea corecta (maze_in = 0) =>
       - o sa se pastreze valorile row si col in r_ant si c_ant, pentru a fi folosite ulterior
       - se activeaza scrierea (maze_we = 1)
       - avanseaza la **3** (next_state = `perete)
    2. directia este gresita (maze_in = 1) =>
       - o sa se intoarca la pozitia initiala (row = r_ant, col = c_ant)
       - dir se incrementeaza pentru a schimba directia (ex: dir = 1 => o sa incerce sa mearga in jos)
       - se intoarce in starea **1** pentru a gasi directia potrivita

- **3: PERETE**
  - pentru cele 4 directii se atribuie corespunzator peretele derpt (daca ne uitam la omulet de sus observam ca directiei sus (row - 1) ii corespunde directia stanga (col + 1) ca perete drept) =>
    - 0: sus (row - 1) -> stanga (col + 1)
    - 1: jos (row + 1) -> dreapta (col - 1)
    - 2: dreapta (col - 1) -> sus (row - 1)
    - 3: stanga (col + 1) -> jos (row + 1)
  - aceste noi directii atribuite se vor folosi pentru a verifica daca

exista perete la derapta pe noua pozitie in care se afla omuletul => se activeaza citirea
  - urmeaza **4** (next_state = `testare_perete)

- **4: TESTARE_PERETE**
  - analizam 2 cazuri:
    1. exista perete in dreapta (maze_in = 1)
       - o sa se intoarca in pozitia anterioara (row = r_ant, col = c_ant)
       - o sa se uite in directia de mers, (revine la cele 4 directii prezentate in starea **1**; ex: cazul 0: sus (row - 1)**[DIRECTIE]** -> stanga (col + 1)**[PERETE]** -> **sus (row - 1) [TESTARE_PERETE]**) pentru a-si continua deplasarea
       - se va activa citirea (in urmatoarea stare se doreste verificarea posibilitatii deplasarii in fata)
       - continua in **5** (next_state = `verificare_blocaj)
    2. nu exista perete in dreapta (maze_in = 0)
       - daca in dreapta nu este perete => automat omuletul o sa mearga la dreapta
       - salvam pozitia curenta (r_ant = row, c_ant = col)
       - se activeaza scrierea
       - se verifica daca omuletul se afla la marginea labirintului (daca row sau col sunt 0 sau 63), daca da => se va trece in starea **7** (next_state = `stop), daca nu => se continua
       - directia peretelui o sa devina directia de mers (cazul directiei peretelui drept va deveni cazul implicit; ex: **cazul 0: sus (row - 1)**, cu peretele drept **stanga (col + 1)** corespunzator **cazului 3**, intrucat nu exista perete la dreapta se doreste schimbarea directiei **sus** cu directia **stanga** pentru a putea continua deplasarea => **dir = 3**, similar pentru restul: daca dir = 1 => **dir = 2**, daca dir = 2 => **dir = 0**, daca dir = 3 => **dir = 1**)
       - o sa se intoarca in starea **3** pentru a verifca prezenta peretelui drept in noua pozitie

- **5: VERIFICARE_BLOCAJ**
  - 2 cazuri:
    1. poate sa mearga in fata (maze_in = 0)
       - salvam pozitia curenta (r_ant = row, c_ant = col)
       - se activeaza scrierea
       - se verifica daca omuletul se afla la marginea labirintului (daca row sau col sunt 0 sau 63), daca da => se va trece in starea **7** (next_state = `stop), daca nu => se intoarce in starea **3** pentru a verifca prezenta peretelui drept in noua pozitie

2. nu poate sa mearga in fata (maze_in = 1)
    - o sa se intoarca in pozitia anterioara (row = r_ant, col = c_ant)
    - o sa se uite spre stanga (singura directie ramasa pentru a incerca sa avanseze -> in dreapta este perete, intrucat nu se poate ajunge in aceasta stare daca peretele drept nu exista; nu se doreste sa mearga inapoi daca nu trebuie)
    - directia dir va fi salvata in variabila dir_aux (pentru a putea reveni la ea in cazul in care omuletul se gaseste in fundatura) -> se putea face si fara, dar mi-a fost mie mai usor asa
    - similar cu **TESTARE_PERETE(2)**, stanga o sa devina directia de mers (dir = 0 => **dir = 2**, dir = 1 => **dir = 3**, dir = 2 => **dir = 1**, dir = 3 => **dir = 0**)
    - se va activa citirea, pentru a putea verifica daca in stanga este sau nu perete in urmatoarea stare
    - urmeaza starea **6** (next_state = `fundatura)

- **6: FUNDATURA**
  - 2 cazuri:
    1. poate sa mearga la stanga (maze_in = 0)
        - salvam pozitia curenta (r_ant = row, c_ant = col)
        - se activeaza scrierea
        - se verifica daca omuletul se afla la marginea labirintului (daca row sau col sunt 0 sau 63), daca da => se va trece in starea **7** (next_state = `stop), daca nu => se intoarce in starea **3** pentru a verifca prezenta peretelui drept in noua pozitie
    2. nu poate sa mearga la stanga (maze_in = 1) => merge inapoi
        - o sa se intoarca in pozitia anterioara (row = r_ant, col = c_ant)
        - directia dir va fi directia sa initiala (inainte sa se uite spre stanga -> dir = dir_aux)
        - directia de mers va fi “inapoi” (daca directia sa era **sus (row - 1)** acum va merge in **jos (row + 1)**) -> dir = 0 => **dir = 1**, dir = 1 => **dir = 0**, dir = 2 => **dir = 3**, dir = 3 => **dir = 2**
        - se intoarce in starea **3** pentru a verifca prezenta peretelui drept in noua pozitie (nu este o noua pozitie, ci mai mult o noua perspectiva de mers)

- **7: STOP**
  - omuletul a iesit din labirint -> done = 1