Coursework on Discrete Event Systems

Koral Hassan
Imperial College London
Kensington, London, SW7 2AZ
kbh15@ic.ac.uk

1. Modeling the Map

Firgure 2 illustrates a model G_M of the spatial information provided in Figure 1. Each room corresponds to a state of the automaton. Transitions between states are only allowed when two rooms are next to each other by sharing a wall. In particular, 4 types of events are allowed: n,s,e,w, representing a transition from a room to the next one to the north, south, east and west respectively.

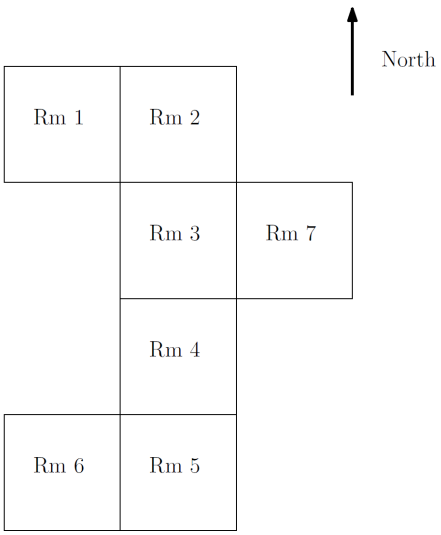

Figure 1. Map of indoor environment.

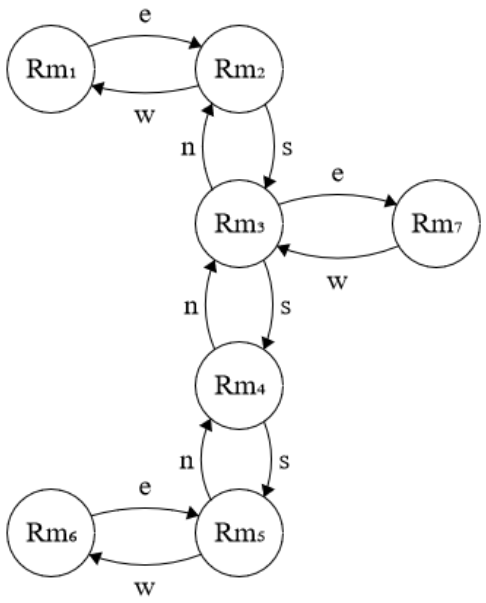

Figure 2. Finite deterministic automaton G_M.

2. Modeling the Robot

Figure 3 illustrates a model G_R of a robot that is able to move in the environment shown in Figure 1, by performing two types of actions;

- rotating its heading clockwise by 90 degrees without changing its position (this happens whenever an r event occurs),
- or moving forward by one unit of space (equal to the size of the room), whenever events n,s,e or w take place.

The automaton keeps track of the robot heading (which is always pointing towards one cardinal direction), making sure that n,s,e or w events may only occur when the heading of the robot is oriented according to the direction of the movement.

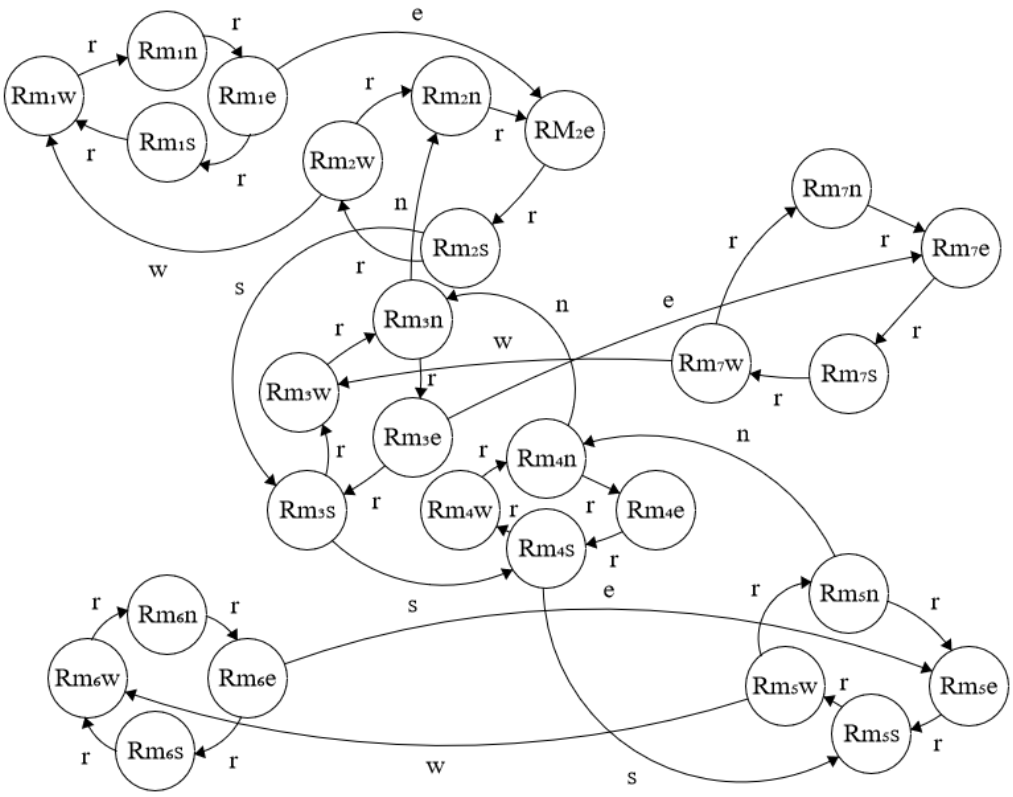

Figure 3. Finite deterministic automaton G_R.

3. Modeling the Robot Inside the Map

3.1. Constructing Automatons

We wish to represent automatons in Matlab by using 4 variables: a list of events, a list of states, a matrix representing a list of transitions and an initial state. Appendix A has the code for a structure that has been created for this purpose.

First, we build G_M and G_R by using the code shown in Appendix B. The name of the events and states have a 1-to-1 correspondance with the lists that represent them. The transition matrix has dimensions $3 \times T$ where T is the number of transitions. Each row represents a transition where the first row is the (index of the) current state, the second row is the (index of the) destination state and the third row is the (index of the) event.

3.2. Parallel Composition

Next, we implement the parallel composition operator. The implementation can be seen in Appendix C We can now model the behavior of the robot inside the environment by building the automaton $G_M||G_R$.

Appendices

A. Constructor for Automata

```
function a = create_automaton(states, events, trans, init, marked, forbidden)
% create_automaton  Constructor for automata
    a.states = states;
    a.events = events;
    a.trans = trans;
    a.init = init;
    if nargin > 4
        a.marked = marked;
    else
        a.marked = states;
    end
    if nargin > 5
        a.forbidden = forbidden;
    else
        a.forbidden = {};
end
```

B. Building G_M and G_R

```
states = char('r1','r2','r3','r4','r5','r6','r7');
events = char('n','e','s','w');
trans = [
    1, 2, 2;
    2, 1, 4;
    2, 3, 3;
    3, 2, 1;
    3, 4, 3;
    3, 7, 2;
    4, 3, 1;
    4, 5, 3;
    5, 4, 1;
    5, 6, 4;
    6, 5, 2;
    7, 3, 4;
    ];
init = 1;
G_m = create_automaton(states, events, trans, init)

states = char('r1n','r1e','r1s','r1w', 'r2n','r2e','r2s','r2w', 'r3n','r3e','r3s',
    'r3w', 'r4n','r4e','r4s','r4w', 'r5n','r5e','r5s','r5w', 'r6n','r6e','r6s','r6w
    ', 'r7n','r7e','r7s','r7w');
events = char('n','e','s','w', 'r');
trans = [
    %Room 1
    1, 2, 5;
    2, 3, 5;
    3, 4, 5;
    4, 1, 5;

    2, 6, 2;


    %Room 2
    5, 6, 5;
    6, 7, 5;
    7, 8, 5;
    8, 5, 5;

    7, 11, 3;
    8, 4, 4;


    % Room 3
    9, 10, 5;
    10, 11, 5;
    11, 12, 5;
    12, 9, 5;

    9, 5, 1;
    10, 26, 2;
```

```
    11, 15, 3;


    %Room 4
    13, 14, 5;
    14, 15, 5;
    15, 16, 5;
    16, 13, 5;

    13, 9, 1;
    15, 19, 3;


    % Room 5
    17, 18, 5;
    18, 19, 5;
    19, 20, 5;
    20, 17, 5;

    17, 13, 1;
    20, 24, 4;


    % Room 6
    21, 22, 5;
    22, 23, 5;
    23, 24, 5;
    24, 21, 5;

    22, 18, 2;


    % Room 7
    25, 26, 5;
    26, 27, 5;
    27, 28, 5;
    28, 25, 5;

    28, 12, 4;
    ];
init = 1;
G_r = create_automaton(states, events, trans, init)
```

C. Parallel Composer

```
function a = parallel_composition(aut1, aut2)
% create_automaton  Constructor for automata
    a.aut1 = aut1;
    a.aut2 = aut2;

    a.states = allcomb(aut1.states, aut2.states);

    for i = length(aut2.events)
        if ismember(aut2.events(i), aut1.events)
            tmp = aut2.trans(:,3);
            aut1_event_loc = find(aut2.events(i));
            tmp(tmp==i) = aut1_event_loc(1);
            aut2.trans(:,3) = tmp;
        else
            aut1.events = [aut1.events aut2.events(i)]
        end
    end

end
```