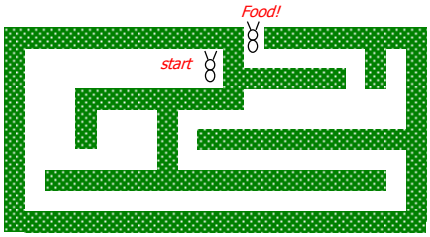
Lecture 19

- A bigger FSM example: Hungry Robot Ant in Maze

Robotic ant in a maze

- Help the ant escape the maze to find food!
  - Maze has no islands

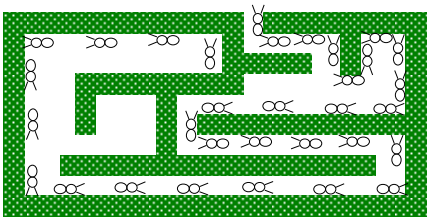


Robotic ant specifics

- Sensors
  - L and R antennae, 1 if touching wall
- Actuators
  - F - forward step, TL/TR - turn left/right
- What strategy should we use?
  - Right-hand rule! (It works in physics and pretty much everywhere else…)

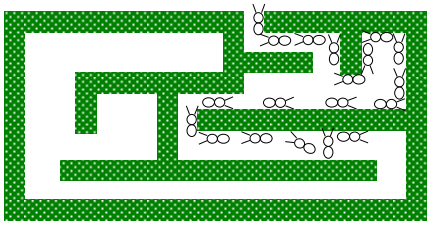
Right-hand rule

- Keep the wall to the right of the ant



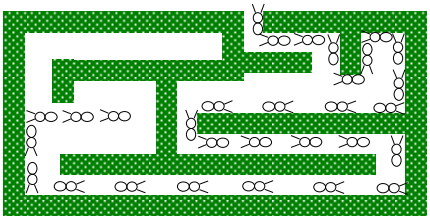
Special cases: What to do?

- Left (L) antenna touching the wall



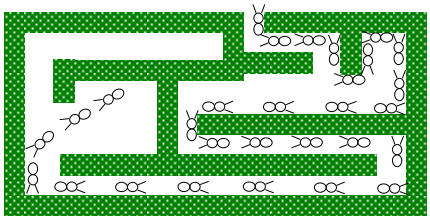
Special cases: What to do?

- Ant lost (not adjacent to any wall)

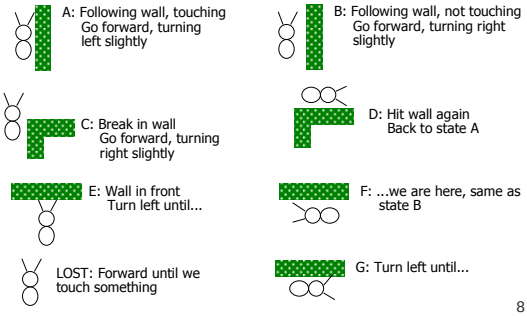


Special cases: What to do?

- Ant lost (not adjacent to any wall)



Robot ant behavior



Notes and strategy

- Notes
  - Maze has no islands
  - Corridors are wider than ant
  - Don't worry about startup
  - Assume a Moore machine
  - Assume D flip-flops
- Strategy
  - Right-hand rule!

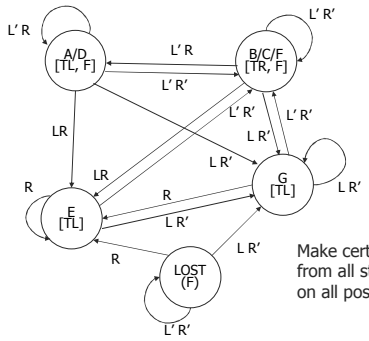
FSM design procedure

1. State diagram
2. State-transition table
3. State minimization
4. State encoding
5. Next-state logic minimization
6. Implement the design

Notations

- Sensors on L and R antennae
  - Sensor = "1" if touching wall; "0" if not touching wall
    - L'R' ≡ no wall
    - L'R ≡ wall on right
    - LR' ≡ wall on left
    - LR ≡ wall in front
- Movement
  - F ≡ forward one step
  - TL ≡ turn left slightly
  - TR ≡ turn right slightly

1. State diagram



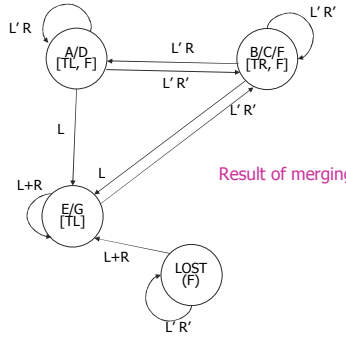
Make certain that transitions from all states are defined on all possible inputs

2. State transition table

state	L	R	next state	outputs
LOST	0	0	LOST	F
LOST	X	1	E	F
LOST	1	0	G	F
E	0	0	B	TL
E	X	1	E	TL
E	1	0	G	TL
G	0	0	B	TL
G	X	1	E	TL
G	1	0	G	TL
A	0	0	B	TL, F
A	0	1	A	TL, F
A	1	0	G	TL, F
A	1	1	E	TL, F
B	0	0	B	TR, F
B	0	1	A	TR, F
B	1	0	G	TR, F
B	1	1	E	TR, F

Same outputs and transitions
⇒ can merge states

3. State minimization



Result of merging states E and G

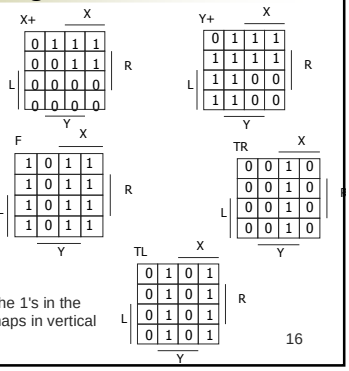
4. State encoding

state	L	R	next state	outputs
LOST	0	0	LOST	F
LOST	X	1	E	F
LOST	1	0	E	F
E	0	0	B	TL
E	X	1	E	TL
E	1	0	E	TL
A	0	0	B	TL, F
A	0	1	A	TL, F
A	1	X	E	TL, F
B	0	0	B	TR, F
B	0	1	A	TR, F
B	1	X	E	TR, F

state		L	R	next state		outputs		
X	Y			X+	Y+	F	TR	TL
0	0	0	0	0	0	1	0	0
0	0	X	1	0	1	1	0	0
0	0	1	0	0	1	1	0	0
0	1	0	0	1	1	0	0	1
0	1	X	1	0	1	0	0	1
0	1	1	0	0	1	0	0	1
1	0	0	0	1	1	1	0	1
1	0	0	1	1	0	1	0	1
1	0	1	X	0	1	1	0	1
1	1	0	0	1	1	1	1	0
1	1	0	1	1	0	1	1	0
1	1	1	X	0	1	1	1	0

5. Next-state logic minimization

state		L	R	next state		outputs		
X	Y			X+	Y+	F	TR	TL
0	0	0	0	0	0	1	0	0
0	0	X	1	0	1	1	0	0
0	0	1	0	0	1	1	0	0
0	1	0	0	1	1	0	0	1
0	1	X	1	0	1	0	0	1
0	1	1	0	0	1	0	0	1
1	0	0	0	1	1	1	0	1
1	0	0	1	1	0	1	0	1
1	0	1	X	0	1	1	0	1
1	1	0	0	1	1	1	1	0
1	1	0	1	1	0	1	1	0
1	1	1	X	0	1	1	1	0



Why are the 1's in the last 3 K-maps in vertical columns?

6. Implement the design

- Outputs are a function of the current state only - Moore machine