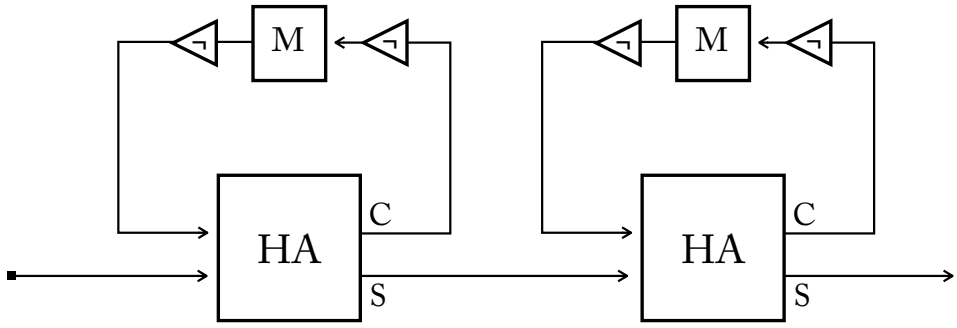
HW4. Computing with FSMs

11.11.25

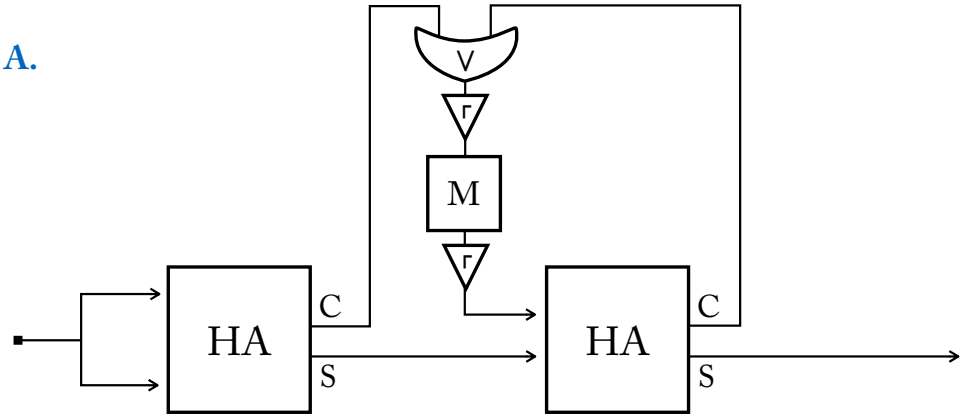
I. State Abstraction

1. **State abstraction.** Construct a state table for the following machine, then convert it to a FSM diagram. What function does it compute in binary?

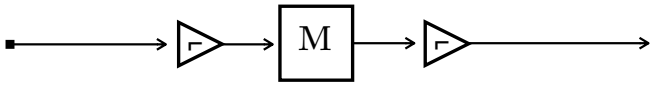


2. **Multiple realizability.** Consider the following two SCs. What function does each compute in binary? Do they implement the same FSM? Why or why not? Show your work.

A.



B.



Design FSMs that compute the following functions.

3. +1 T
4. +2 T
5. +1 B
6. +2 B
7. +4 B
8. $2x$ B
9. $2x$+1 B
10. $2(x+1)$ B
11. $x + y$ B

Advice!

Don't design an SC first to get your solution; **do** think through the FSM directly.

Caution!

Function composition **does not work** for FSMs. You can't combine two +1 machines to get a +2 machine.

This has to do with the fact that FSM diagrams show states of the *overall* machine, not the flow of information *through* it.

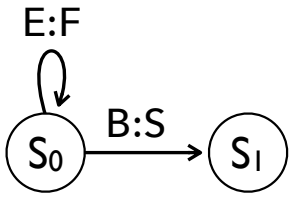
For each of the following navigations tasks, design an FSM which solves it.

Use the following abbreviations:

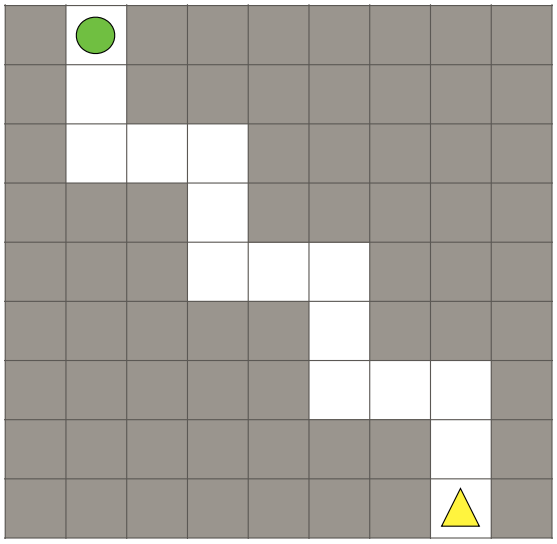
Inputs:
0 = **E** (Empty)
1 = **B** (Block)

Outputs:
11 = **F** (Forward)
00 = **S** (Stop)
10 = **R** (Right)
01 = **L** (Left)

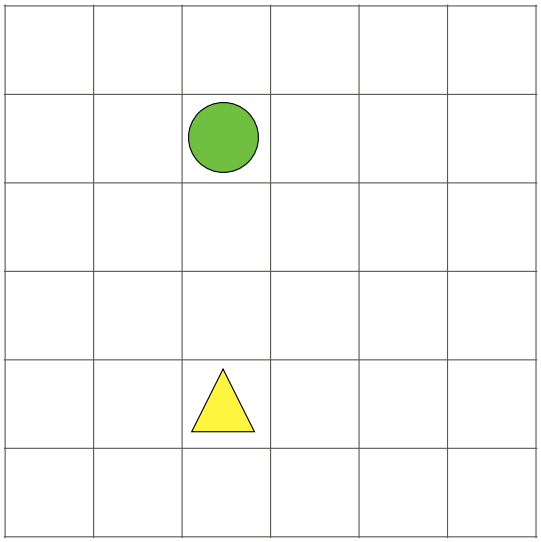
For example, this machine goes forward until it hits a block, then stops:



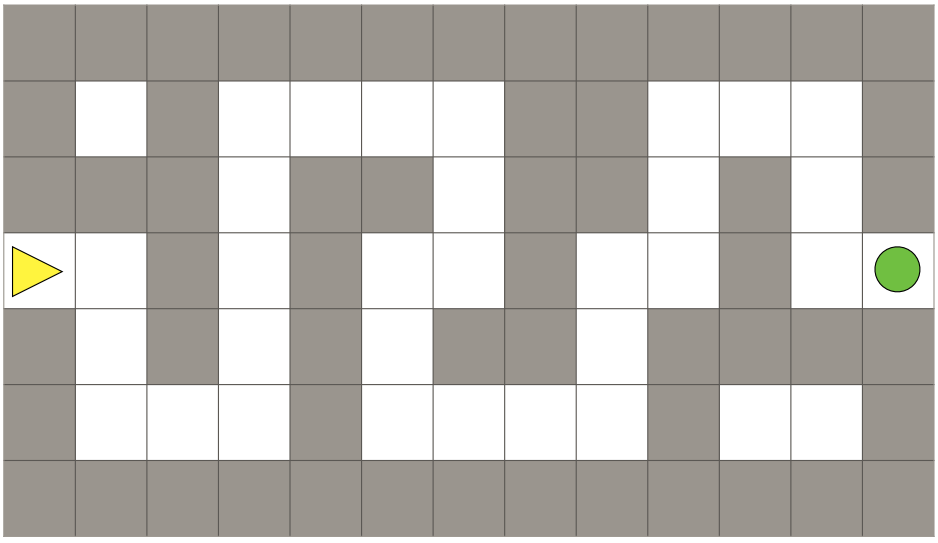
12. **Zig-Zag.** Reach the goal and **keep going**.



13. **Three Ahead.** Reach the goal and **stop**.



14. **Way Finder.** Given *any* non-branching maze, make your way all the way through, and keep going.
A **non-branching maze** is a path that never branches and is never wider than 1-cell across, except on corners where two 1-cell width paths may intersect. Here is an example.*



* This problem was designed by Isaijah Johnson and Jack Schaeffer, Phil 133 W19.