Building Finite State Machines

Murray Cole

Designing FSMs

Given some reactive system, how can build an FSM to model it?

- From scratch, by "intuition", in one go. OK for small examples.
- Build smaller FSMs for parts of the system, then compose them.

How can we convince ourselves that we have got it right?

- By testing on typical cases (beware of "proof by example"!)
- By direct "proof", *i.e.* sound, convincing arguments.
- By proving the smaller parts and our composition methods.

An Example

Design an acceptor FSM with $\Sigma = \{\texttt{0}, \texttt{1}\}$ which accepts only sequences for which the difference between the number of `0`s and the number of `1`s seen at any point during the input never exceeds 1.

- Looks like counting (bad news), but in fact only need to count surplus of `0`s or `1`s until it is larger than 1 either way.
- Need states for "one more `1`", "equal", "one more `0`" and "failed"
- the first three of these are accepting states
- the second is the start state (*i.e.* no input seen, so numbers equal)

Why is this right?

1. Check for all possible inputs of length 0, 1, and 2.

2. Suppose correct for sequences of length n

   - finish in state 1 means one more `0` than `1`
   - finish in state 2 means equal numbers of `0` and `1`
   - finish in state 3 means one more `1` than `0`
   - finish in state 4 means too many `0` or `1` at some point

3. Consider any sequence of length $n + 1$. It is a sequence of length n followed by an extra `0` or `1`. Suppose the length n part leads to state 1. Then, by our assumption above, we have seen one extra `0`. So,
   - if the final input is a `0`, we go to state 4 and don't accept
   - if the final input is a `1`, we go to state 2, and accept

4. Make similar arguments from other states (2, 3, 4).

5. We have now shown that if the machine is correct for inputs of length n, then it must also be correct for inputs of length $n + 1$.

6. Already shown correct for inputs of length 0, 1, 2. Must be true for inputs of length 3. And 4. And 5.....

Composing FSMs

Build large FSMs by building smaller FSMs for parts of the system, then compose them.

Think about acceptors only

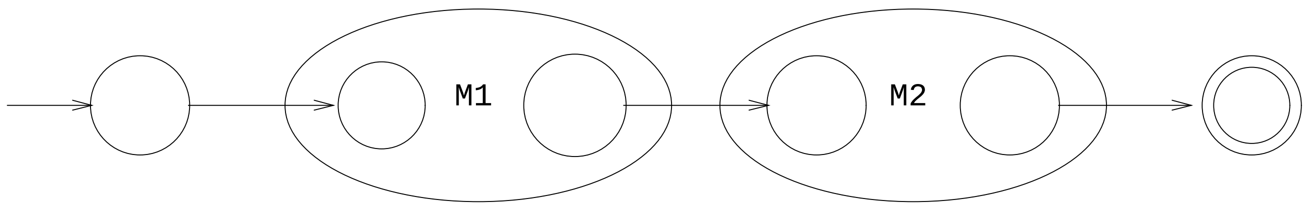
Many kinds of composition:

- Sequence
- Choice
- Repeat
- Intersection

Sequencing

Aim is to produce machine $M1M2$ accepting inputs which have a section accepted by $M1$ followed by a section accepted by $M2$

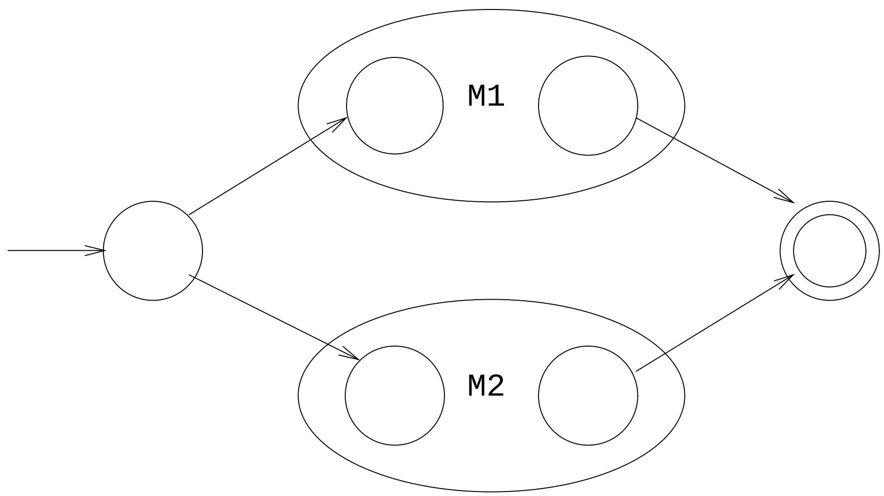
Assume exactly one finish state in each of $M1, M2$ (easy to fix if not)



- new start and accept states
- old accept states now ordinary
- only add ϵ transitions

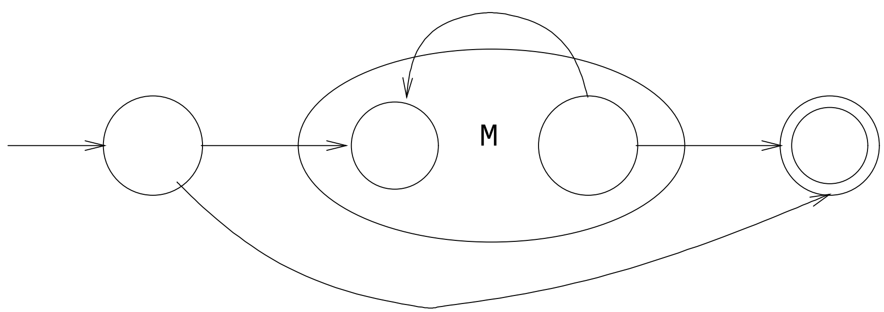
Choice

Aim is to produce machine $M1 \mid M2$ accepting inputs which are accepted by $M1$ or by $M2$ (or by both)



Repetition

Aim is to produce machine M^* accepting inputs which consist of zero or more sections, each individually accepted by M

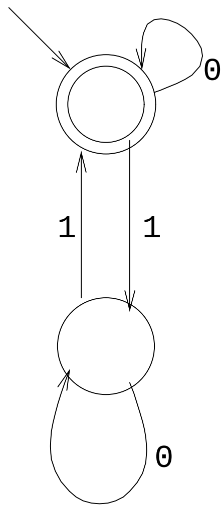


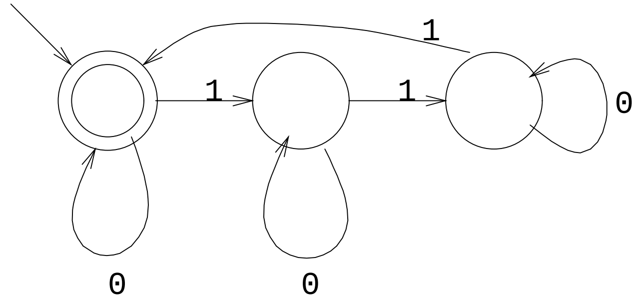
Intersection

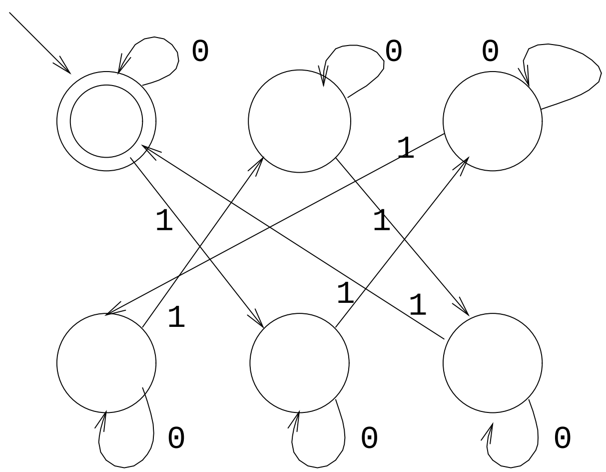
Aim is to produce machine $M1 \cap M2$ accepting inputs which are accepted by $M1$ and by $M2$ (no internal ϵ)

Need to track both machines simultaneously!

- New machine states represent pairs of states from M_1, M_2.
- Start state is the pair of the start states of M_1 and M_2.
- Accepting state is the pair of the accepting states of M_1 and M_2.
- There is a transition labelled a in the new machine between state (p, q) and (r, s) just when there is a transition labelled a between p and r in M_1 *and* a transition labelled a between q and s in the M_2.


0
1
1
0

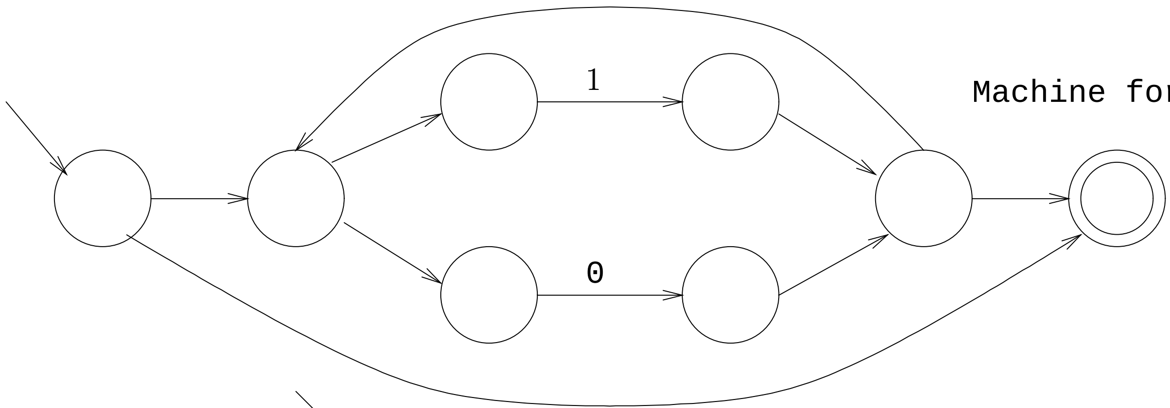

1
1
1
0
0
0


0
0
0
1
1
1
1
1
1
0
0
0

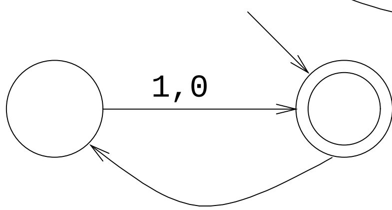
An Example

Design an acceptor FSM with $\Sigma = \{0, 1\}$ which accepts sequences containing two successive 0s and two successive 1s.

- This is $((0 \mid 1)^*11(0 \mid 1)^*) \cap ((0 \mid 1)^*00(0 \mid 1)^*)$
- First a machine for $(0 \mid 1)^*$
- Build into a machine for $((0 \mid 1)^*11(0 \mid 1)^*)$ and simplify it
- Machine for "two successive 0s" is very similar
- Use the intersection construction to complete the task

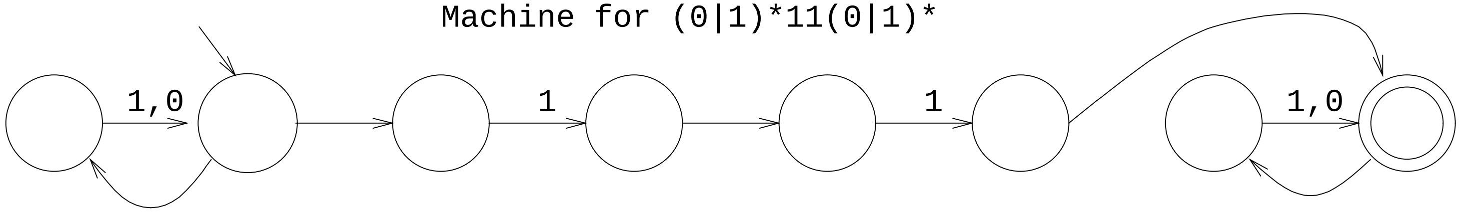


Machine for (0|1)*

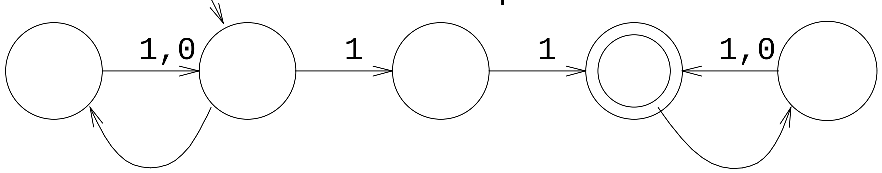


Simplified version

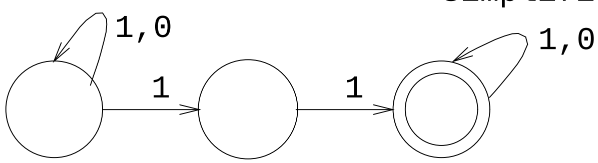
Machine for (0|1)*11(0|1)*

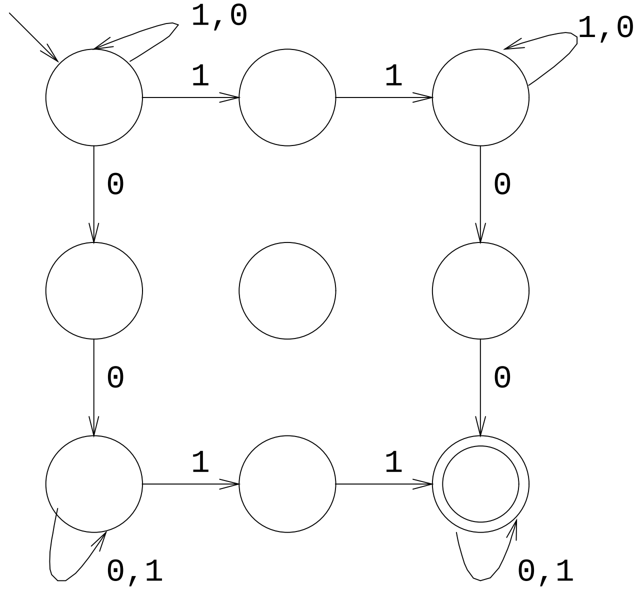


Simplified version



Simplified to eliminate epsilon transitions





An acceptor FSM with $\Sigma = \{0, 1\}$ which accepts sequences containing two successive `0`s and two successive `1`s.

Summary

- Designing and "proving" FSM's from scratch

- Designing FSMs by correct composition of correct simpler FSMs
  - Sequence
  - Choice
  - Repeat
  - Intersection
  - (Interleaving)