Finite state machines

- Set of states
- Set of transitions
- Set of triggers for transitions
- Optional actions

- What are the triggers?
- Typically input data is sequentially processed

- Tokenizing strings, decoding grammars, electrical simulations...

Flavours

- States can be:
  - Start states
  - Final or “accepting” states
- Accepting states can be used to detect particular patterns
  - Recognizers
- Transducers can generate output using actions at any stage
  - May rely on pre, post and current input to generate different outputs
- Markov models...

How to encode?

- Use predicates:
  - to detail structure
  - permissible transitions
  - triggers for transitions

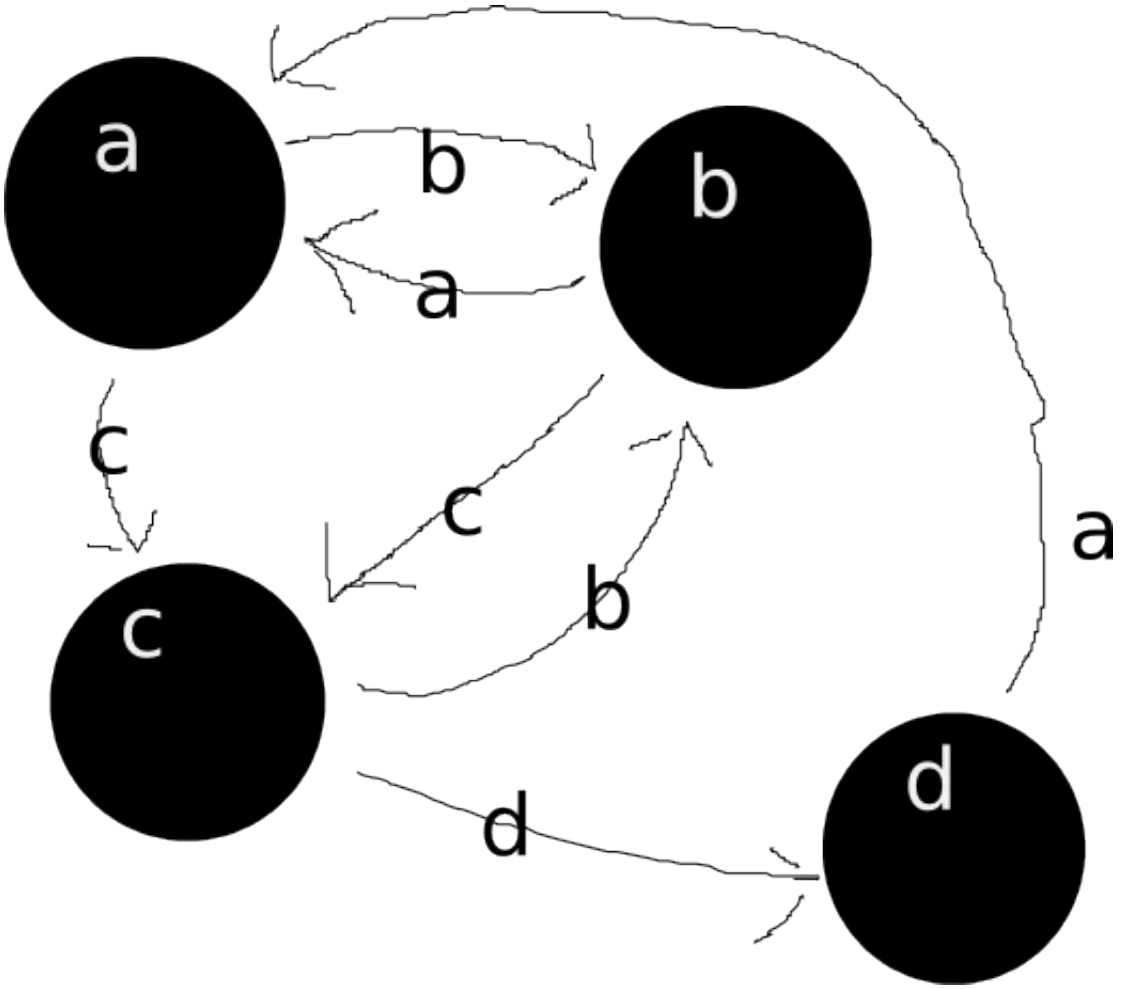
```
move(fromstate, trigger, tostate).
```

- Find a way to process some input data to generate triggers

example

```
move (a, char_b, b).
move (a, char_c, c).
move (b, char_a, a).
move (b, char_c, c).
move (c, char_d, d).
move (c, char_b, b).
move (d, char_a, a).
```

example



How to traverse?

- Lots of ways to do it
- Easiest ways use lists to represent input
  - Lists represented using square brackets
  - Can split lists into head and tail

```
head([H|T],H)

head([a,b,c,d,e],X)
```

- Representing input string as a list allows traversal of the FSM to be treated as a list processing problem

How to traverse?

```
traverse([Token|R], Current, Final):-
        move(Current, Token, Next),
        traverse(R, Next, Final).

traverse([], Final, Final).
```

- Take item from head of list
- Make move to new state
- Call recursively until end of list

Turing Machine?

- How might you implement one?
- How might you represent the tape?
- Need to have some notion of actions associated with particular states:
  - Move tape
  - Mark tape