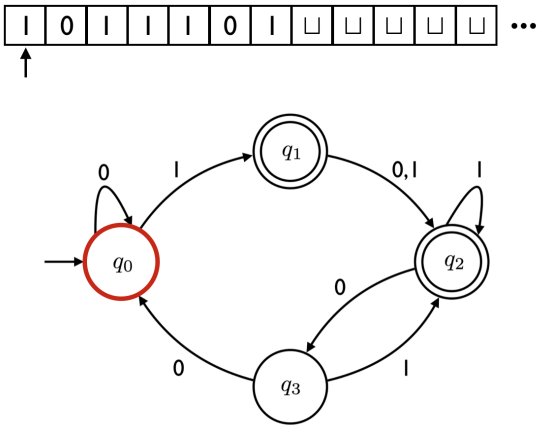
Goal of this lecture:

Goal of next lecture:

DFA: state diagram + input tape



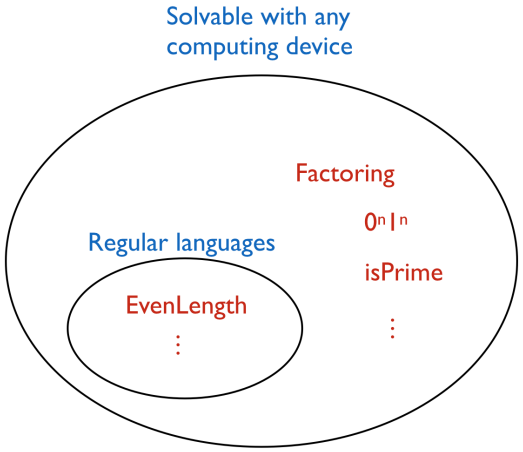
DFA: code

```
def foo(input):
  i = 0;
  STATE 0:
    if (i == input.length): return False;
    letter = input[i];
    i++;
    switch(letter):
      case '0': go to STATE 0;
      case '1': go to STATE 1;

  STATE 1:
    if (i == input.length): return True;
    letter = input[i];
    i++;
    switch(letter):
      case '0': go to STATE 2;
      case '1': go to STATE 2;
  ...
```

We would like to define computation/algorithm mathematically formally.
The properties we want from this definition are:

2 Important Observations:



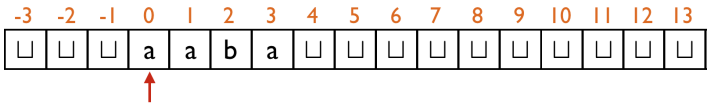
Should we define computable to mean what is computable by a Python function/program?

Downsides of this:

So what we want is:

Turing Machine Description

TM ≈ DFA + infinite tape



TM could have been defined as a sequence of instructions, where the allowed instructions are:

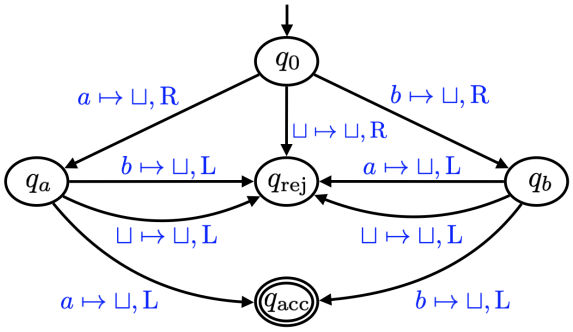
- Move the head left
- Move the head right
- Write a symbol a (from the alphabet)
- If head is reading symbol a, GOTO step j
- Halt and accept
- Halt and reject

But even this is not simple enough. We encapsulate these into one instruction:

STATE 0:
  switch(letter under the head):
    case 'a': **write** 'b'; **move** Left; **go to** STATE 2;
    case 'b': **write** '␣'; **move** Right; **go to** STATE 0;
    case '␣': **write** 'b'; **move** Left; **go to** STATE 1;

At each step you have to:

Don't want to change the symbol:
Want to stay put:
Don't want to change state:




```
def M(input):
  i = 0
  STATE 0:
    letter = input[i];
    switch(letter):
      case 'a':  input[i] = ' '; i++; go to STATE a;
      case 'b':  input[i] = ' '; i++; go to STATE b;
      case ' ':  input[i] = ' '; i++; go to STATE rej;
  STATE a:
    letter = input[i];
    switch(letter):
      case 'a':  input[i] = ' '; i--;  go to STATE acc;
      case 'b':  input[i] = ' '; i--;  go to STATE rej;
      case ' ':  input[i] = ' '; i--;  go to STATE rej;

  ⋮
```

The machine accepts a string x if and only if:

Turing Machine Formal Definition

A Turing machine (TM) M is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, q_{\text{acc}}, q_{\text{rej}})$$

where

-
-
-
-
-
-
-

TMs vs DFAs

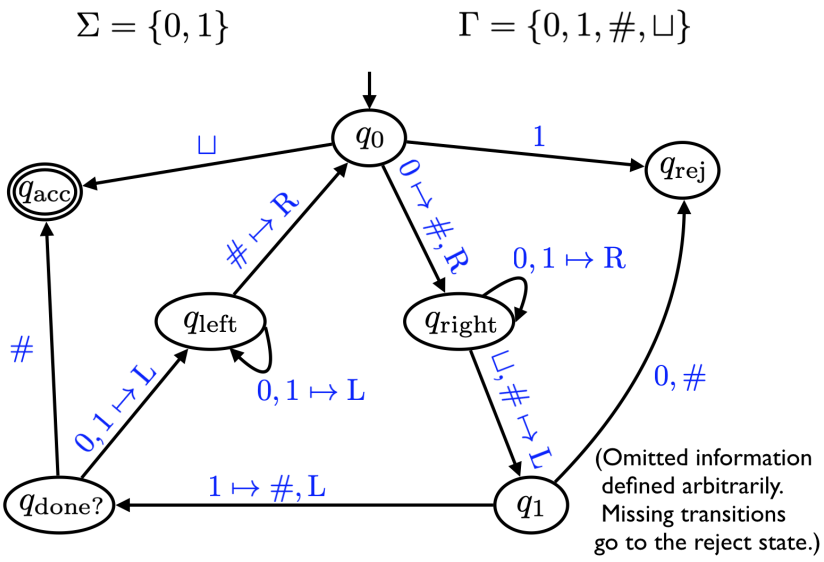
-
-
-
-
-

What is the analog of regular languages in this setting?

Decider:

Decidable language:

Irregular but decidable: Turing machine for $\{0^n 1^n : n \in \mathbb{N}\}$.



Some TM subroutines and tricks

- Move right (or left) until first $\sqcup$ encountered.
- Shift entire input string one cell to the right.
- Convert input from $x_1 x_2 \dots x_n$ to $\sqcup x_1 \sqcup x_2 \sqcup x_3 \dots \sqcup x_n$.
- Simulate a big Γ with just $\{0, 1, \sqcup\}$.
- Mark cells. If $\Gamma = \{0, 1, \sqcup\}$, extend it to $\Gamma = \{0, 1, 0', 1', \sqcup\}$.
- Copy a stretch of tape between two marked cells into another marked section of the tape.
- Implement basic string and arithmetic operations.
- Simulate a TM with 2 tapes and heads.
- Implement basic data structures.
- Simulate "random access memory".
- $\dots$
- Simulate assembly language.

3 ways to describe a TM

1. Low level description:

2. Medium level description:

3. High level description:

Important Question

Is TM the right definition?