Main Question
What problems can a particular type of machine solve?

What is a problem?
What is a machine?
What does it mean for a machine to solve a problem?

Problems Defining Problem

A problem is defined by:
– A description of a set of inputs
– A description of a set of outputs and the property an output must have

A machine solves a problem if for every input it eventually produces a satisfactory output.

Finite Problems
Uninteresting: can be solved by a lookup machine

Problems with a finite number of possible inputs

All theoretically interesting problems have infinitely many possible inputs.

Outputs

How many possible outputs do you need for a problem to be interesting?

2 – “Yes” or “No”

A decision problem is a problem that has two possible outputs.

In the real world lots of interesting problems have finite number of inputs: chess (0 or 1 inputs?), Internet search with bounded-length query, human genome assembly, etc.
Language Recognition

Is string $s$ in language $L$?

A machine $M$ recognizes language $L$ if it can solve: for any string $s$, is $s$ in $L$?

We can describe the power of a type of machine is by the set of languages it can recognize.

Deterministic Finite Automata

A finite set of states
Set of accepting states
One start state
Transition function

DFA Example

Draw a DFA that recognizes the language of strings in $[0, 1]^*$ with an even number of 1s.

Recap: What is a language?
Recap: What does it mean to recognize a language?

DFA Design Tips

- Make sure you understand the target language: think of example strings in and not in the language, don’t forget about $\varepsilon$
- Think about what the states represent (e.g., what is the current remainder)
- Walk through what the machine should do on example inputs (both accepting and rejecting)

“Trick” Question

What languages can be recognized by a DFA?

The regular languages.

This is the definition of a regular language: a language is a regular language if there is some DFA that recognizes it.

“Tricky” Questions?

Can all finite languages be recognized by a DFA?

Yes. Trivially: create a state-path for each string in the language. Finite language, means a finite number of states is enough.

Can a DFA recognize an infinite language?

Yes. We’ve seen two examples already!

Formal Definition

A finite automaton is a 5-tuple:

$Q$ finite set (“states”)
$\Sigma$ finite set (“alphabet”)
$\delta : Q \times \Sigma \to Q$ transition function
$q_0 \in Q$ start state
$F \subseteq Q$ set of accepting states
Computation Model

Define $\delta^*$ as the extended transition function:

$$w \in \mathcal{L}(A) \Leftrightarrow \delta^*_A(q_0, w) \in F_A$$

A string, $w$, is in the language defined by DFA $A$ iff the result of applying the extended transition function of $A$ to start state, $q_0$, and $w$ is a final state.

If you prefer Java code...

```java
State nextState(State q, String w) {
    if (w.length() == 0)
        return q;
    else
        return (nextState(transition(q, w.charAt(0)), w.substring(1)));
}
```

Complement

Is the set of regular languages is closed under complement?

**Proof.** By definition, if $L$ is a regular language there exists some DFA $M = (Q, \Sigma, \delta, q_0, F)$ that recognizes $L$.

We prove that $\overline{L}$ is regular by showing how to construct a DFA $\overline{M}$ that recognizes $\overline{L}$.

$$\overline{M} = (Q, \Sigma, \delta, q_0, F')$$

where $F' = Q - F$

$\overline{M}$ accepts every string that is rejected by $M$ since the accepting states of $\overline{M}$ are the rejecting states of $M$ and vice versa.

$$w \in L \Leftrightarrow \delta^*(q_0, w) \in F$$

$$w \in \overline{L} \Leftrightarrow \delta^*(q_0, w) \notin F \Leftrightarrow \delta^*(q_0, w) \in F'$$

Charge

- **Thursday’s Class:**
  - guest lecture by Gabe Robins
  - office hours (Sonali) in Stacks after class

- **PS1 Due Tuesday** (problem 4 removed)