cs3102: Theory of Computation

Class 2:
Problems and Finite Automata

Spring 2010
University of Virginia
David Evans

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*?

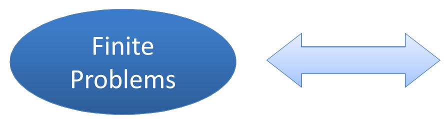
Problems Defining Problem

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

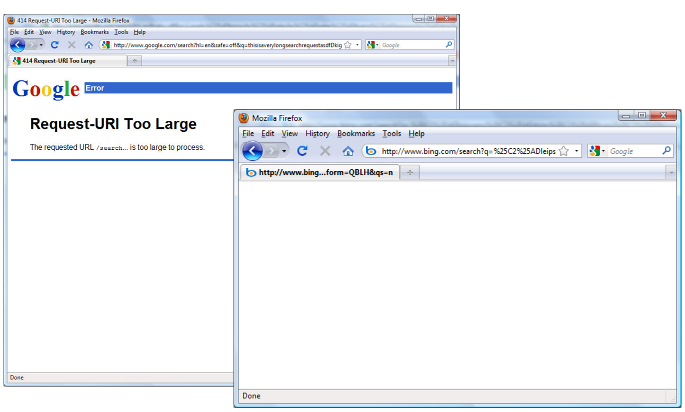
A machine **solves** a problem if for *every* **input** it *eventually* produces a satisfactory **output**.



Problems with a finite number of possible inputs

Uninteresting: can be solved by a lookup machine

All theoretically interesting problems have infinitely many possible inputs.



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.

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.

Finite search problems can be framed as a series of decision problems:

What is **2+2**? vs.	Does **2+2=0**?	**No**
	Does **2+2=1**?	**No**
	Does **2+2=2**?	**No**
	Does **2+2=3**?	**No**
	Does **2+2=4**?	**Yes**

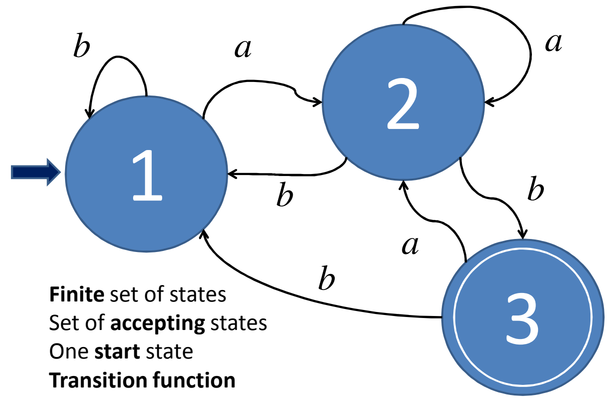
Language Recognition

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



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 ε
- 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")
Σ	**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 δ* 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**.

String (prepending definition)
Basis: ε (the empty string) is a **string**
Induction: if s is a string, and $a \in \Sigma$, as is a string

String (appending definition)
Basis: ε (the empty string) is a **string**
Induction: if s is a string, and $a \in \Sigma$, sa is a string

Computation Model

Define δ* as the **extended transition function:**

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

$\delta^* : Q \times \Sigma^* \rightarrow Q$ Function from a state and string to a state.

Basis: $w = \epsilon$

$$\delta^*(q, \epsilon) = q$$

Induction: $w = ax, a \in \Sigma, x \in \Sigma^*$

$$\delta^*(q, w) = \delta^*(\delta(q, a), x)$$

Defining δ* for the other appending string definition is left as exercise (or exam question?).

If you prefer Java code...

```
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 $\overline{L}$:

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

$\overline{M}$ 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)