CS21 Lecture 2
January 5, 2022

Outline
- problems, languages, machines
- Finite Automata
- Nondeterministic Finite Automata
- Closure under regular operations
- NFA, FA equivalence

What is a problem?
- For most of this course, a problem is a decision problem:
  \( f : \Sigma^* \to \{ \text{accept, reject} \} \)
- Equivalent notion: language
  \( L \subseteq \Sigma^* \)
  the set of strings that map to "accept"
- Example: \( L = \) set of pairs \((m, k)\) for which \( m \) has a prime factor \( p < k \)

What is computation?
- the set of strings that lead to "accept" is the language recognized by this machine
- if every other string leads to "reject", then this language is decided by the machine

Terminology
- finite alphabet \( \Sigma \): a set of symbols
- language \( L \subseteq \Sigma^* \): subset of strings over \( \Sigma \)
- a machine takes an input string and either
  - accepts, rejects, or
  - loops forever
- a machine recognizes the set of strings that lead to accept
- a machine decides a language \( L \) if it accepts \( x \in L \) and rejects \( x \notin L \)

What goes inside the box?
- We want the simplest mathematical formalization of computation possible.
- Strategy:
  - endow box with a feature of computation
  - try to characterize the languages decided
  - identify language we "know" real computers can decide that machine cannot
  - add new feature to overcome limits
Finite Automata

- simple model of computation
- reads input from left to right, one symbol at a time
- maintains state: information about what seen so far ("memory")
  - finite automaton has finite # of states: cannot remember more things for longer inputs
- 2 ways to describe: by diagram, or formally

FA diagrams

- (single) start state
- alphabet $\Sigma = \{0,1\}$
- (several) accept states
- transition for each symbol

- read input one symbol at a time; follow arrows; accept if end in accept state

FA operation

- Example of FA operation:
  - input: 0 1 0 1
  - not accepted

- Example of FA operation:
  - input: 1 0 1
  - accepted

What language does this FA recognize?

$L = \{x : x \in \{0,1\}^*, x_1 = 1\}$

Example FA

- What language does this FA recognize?
  
$L = \{x : x \in \{0,1\}^*, x \text{ has even # of 1s}\}$
- illustrates fundamental feature/limitation of FA:
  - "tiny" memory
  - in this example only "remembers" 1 bit of info.

Example FA

$\Sigma = \{A,B,C\}$

Try:
- AC
- CBCC
- AA
- BBBBBBB
- CCBC

"35 cents"
FA formal definition

A finite automaton is a 5-tuple

\((Q, \Sigma, \delta, q_0, F)\)

- \(Q\) is a finite set called the states
- \(\Sigma\) is a finite set called the alphabet
- \(\delta: Q \times \Sigma \rightarrow Q\) is a function called the transition function
- \(q_0\) is an element of \(Q\) called the start state
- \(F\) is a subset of \(Q\) called the accept states

Formal description of FA operation

finite automaton

\(M = (Q, \Sigma, \delta, q_0, F)\)

accepts a string

\(w = w_1w_2w_3...w_n \in \Sigma^*\)

if \(\exists\) sequence \(r_0, r_1, r_2, ..., r_n\) of states for which

- \(r_0 = q_0\)
- \(\delta(r_i, w_{i+1}) = r_{i+1}\) for \(i = 0,1,2,...,n-1\)
- \(r_n \in F\)

Characterizing FA languages

- We will show that the set of languages recognized by FA is closed under:
  - union "\(C = (A \cup B)\)"
  - concatenation "\(C = (A \circ B)\)"
  - star "\(C = A^*\)"
- Meaning: if \(A\) and \(B\) are languages recognized by a FA, then \(C\) is a language recognized by a FA

What now?

- We have a model of computation
  - (Maybe this is it. Maybe everything we can do with real computers we can do with FA…)
- try to characterize the languages FAs can recognize
  - investigate closure under certain operations
- show that some languages not of this type

Characterizing FA languages

- union "\(C = (A \cup B)\)"
  
  \((A \cup B) = \{x : x \in A \text{ or } x \in B \text{ or both}\}\)

- concatenation "\(C = (A \circ B)\)"
  
  \((A \circ B) = \{xy : x \in A \text{ and } y \in B\}\)

- star "\(C = A^*\)"
  - (note: \(\varepsilon\) always in \(A^*\))
  
  \(A^* = \{x_1x_2x_3...x_k : k \geq 0 \text{ and each } x_i \in A\}\)
Concatenation attempt

\[(A \circ B) = \{xy : x \in A \text{ and } y \in B\}\]

What label do we put on the new transitions?

Nondeterministic FA

- We will make life easier by describing an additional feature (nondeterminism) that helps us to “program” FAs
- We will prove that FAs with this new feature can be simulated by ordinary FA – same spirit as programming constructs like procedures
- The concept of nondeterminism has a significant role in TCS and this course.

NFA diagrams

- At each step, several choices for next state

NFA operation

- Example of NFA operation:

\[
\begin{array}{c}
\text{alphabet } \Sigma = \{0,1\} \\
\begin{array}{c}
\text{states} \\
\text{transitions:}
\end{array}
\end{array}
\]

input: 0 1 0
not accepted

- Example of NFA operation:

\[
\begin{array}{c}
\text{alphabet } \Sigma = \{0,1\} \\
\begin{array}{c}
\text{states} \\
\text{transitions:}
\end{array}
\end{array}
\]

input: 1 1 0
accepted
NFA operation

• One way to think of NFA operation:

• string $x = x_1x_2x_3...x_n$ accepted if and only if
  – there exists a way of inserting $\epsilon$'s into $x$
    $x_1\epsilon\epsilon x_2 x_3...\epsilon x_n$
  – so that there exists a path of transitions from
    the start state to an accept state