Finite Automata

Part Two
Recap from Last Time
Formal Language Theory

• An alphabet is a set, usually denoted Σ, consisting of elements called characters.
• A string over Σ is a finite sequence of zero or more characters taken from Σ.
• The empty string has no characters and is denoted ε.
• A language over Σ is a set of strings over Σ.
• The language Σ* is the set of all strings over Σ.
DFAs

• A \textbf{DFA} is a
  • \textbf{D}eterministic
  • \textbf{F}inite
  • \textbf{A}utomaton

• DFAs are the simplest type of automaton that we will see in this course.
DFAs

- A DFA is defined relative to some alphabet $\Sigma$.
- For each state in the DFA, there must be exactly one transition defined for each symbol in $\Sigma$.
  - This is the “deterministic” part of DFA.
- There is a unique start state.
- There are zero or more accepting states.
A Sample DFA

$L = \{ \ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \ \}$
A Sample DFA

\[ L = \{ w \in \{a, b\}^* | w \text{ contains } aa \text{ as a substring } \} \]
New Stuff!
Tabular DFAs

These stars indicate accepting states.

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>*q_0</td>
<td>q_1</td>
<td>q_0</td>
</tr>
<tr>
<td>q_1</td>
<td>q_3</td>
<td>q_2</td>
</tr>
<tr>
<td>q_2</td>
<td>q_3</td>
<td>q_0</td>
</tr>
<tr>
<td>*q_3</td>
<td>q_3</td>
<td>q_3</td>
</tr>
</tbody>
</table>
Since this is the first row, it's the start state.
Tabular DFAs

Question to ponder: Why isn't there a column here for $\Sigma$?
int kTransitionTable[kNumStates][kNumSymbols] = {
    {0, 0, 1, 3, 7, 1, ...},
    ...
};
bool kAcceptTable[kNumStates] = {
    false,
    true,
    true,
    ...
};
bool SimulateDFA(string input) {
    int state = 0;
    for (char ch: input) {
        state = kTransitionTable[state][ch];
    }
    return kAcceptTable[state];
}
The Regular Languages
A language $L$ is called a regular language if there exists a DFA $D$ such that $\mathcal{L}(D) = L$.

If $L$ is a language and $\mathcal{L}(D) = L$, we say that $D$ recognizes the language $L$. 
The Complement of a Language

- Given a language $L \subseteq \Sigma^*$, the **complement** of that language (denoted $\overline{L}$) is the language of all strings in $\Sigma^*$ that aren't in $L$.
- Formally:

  $$ \overline{L} = \Sigma^* - L $$

Good proofwriting exercise: prove $\overline{\overline{L}} = L$ for any language $L$. 
Complementing Regular Languages

$L = \{ \ w \in \{a, b\}^* \mid w \text{ contains } aa \text{ as a substring} \ \}$

\[ \begin{array}{c}
q_0 \xrightarrow{a} q_1 \\
& \xrightarrow{b} q_0 \\
\end{array} \]

\[ \begin{array}{c}
q_1 \xrightarrow{a} q_2 \\
\end{array} \]

\[ \begin{array}{c}
q_2 \xrightarrow{\Sigma} \end{array} \]

$\bar{L} = \{ \ w \in \{a, b\}^* \mid w \text{ does not contain } aa \text{ as a substring} \ \}$

\[ \begin{array}{c}
q_0 \xrightarrow{b} q_0 \\
& \xrightarrow{a} q_1 \\
\end{array} \]

\[ \begin{array}{c}
q_1 \xrightarrow{a} q_2 \\
& \xrightarrow{b} q_1 \\
\end{array} \]

\[ \begin{array}{c}
q_2 \xrightarrow{\Sigma} \end{array} \]
Complementing Regular Languages

\[ L = \{ w \in \{ a, *, / \}^* \mid w \text{ represents a C-style comment} \} \]
Complementing Regular Languages

\[ \bar{L} = \{ w \in \{a, *, /\}^* \mid w \text{ doesn't represent a C-style comment} \} \]
Closure Properties

- **Theorem:** If $L$ is a regular language, then $\overline{L}$ is also a regular language.
- As a result, we say that the regular languages are **closed under complementation**.

Question to ponder: are the nonregular languages closed under complementation?
NFAs
Revisiting a Problem
NFAs

• An **NFA** is a
  • **N**ondeterministic
  • **F**inite
  • **A**utomaton

• Structurally similar to a DFA, but represents a fundamental shift in how we'll think about computation.
(Non)determinism

- A model of computation is **deterministic** if at every point in the computation, there is exactly one choice that can make.
  - The machine accepts if that series of choices leads to an accepting state.
- A model of computation is **nondeterministic** if the computing machine has zero or multiple decisions that it can make at one point.
  - The machine accepts if **any** series of choices leads to an accepting state.
- (This sort of nondeterminism is technically called **existential nondeterminism**, the most philosophical-sounding term we’ll introduce all quarter.)
A Simple NFA

$q_0$ has two transitions defined on 1!
A More Complex NFA

If a NFA needs to make a transition when no transition exists, the automaton dies and that particular path does not accept.
Hello, NFA!

- Start state $q_0$ transitions on letter 'h' to state $q_1$.
- State $q_1$ transitions on letter 'i' to state $q_2$.
Tragedy in Paradise
The language of an NFA is \( \mathcal{L}(N) = \{ w \in \Sigma^* \mid N \text{ accepts } w \} \).

What is the language of each NFA? (Assume \( \Sigma = \{a, b\} \).)

**Note that flipping the accept and reject states of an NFA doesn’t always give an NFA for the complement of the original language. (Why?)**

**Question to ponder:** Why is the answer \( \{ w \in \Sigma^* \mid w \text{ ends in } \text{aaa} \} \) not correct?

\( \{ w \in \Sigma^* \mid w \text{ ends in } \text{aa} \} \)

\( \emptyset \)
ε-Transitions

• NFAs have a special type of transition called the \textbf{ε-transition}.

• An NFA may follow any number of ε-transitions at any time without consuming any input.

Not at all fun or rewarding exercise: what is the language of this NFA?
ε-Transitions

• NFAs have a special type of transition called the **ε-transition**.

• An NFA may follow any number of ε-transitions at any time without consuming any input.

• NFAs are not *required* to follow ε-transitions. It's simply another option at the machine's disposal.
Time-Out For Announcements!
Midterms Graded

- Midterms have been graded! If you didn’t pick yours up yet, you can grab it from the Gates building.

  - SCPD students – exams have been sent back to the SCPD distribution office. If you haven’t received yours yet, ping the SCPD distribution office.

- We’ve posted a regrade request form on the course website with instructions about how to ask for a regrade. Regrade requests are due next Wednesday.
Your Questions
“Why do the majority of students here neglect to wear their helmets? I always assumed Stanford people would want to protect their heads”

WEAR A BIKE HELMET.
NO EXCUSES.
“In your opinion, as a 20 year old, what is the best reason to be optimistic for the future?”

It’s almost fashionable these days to be fatalistic about the future, and it’s a position I’ve never understood. People are so smart, so clever, and so resourceful that I’m genuinely excited to see what the future has in store for us.

We’ve got a lot of challenges to face, but we have historical precedents we can look to to see how to best resolve them. The particular challenges will change, but we’ll find ways to figure them out. And despite what a lot of folks say, I don’t think we’re facing any crises that are so totally unlike our past ones that we can’t possibly address them. We’ve found lots of creative ways to solve major problems in the past. I’m sure we can do it again.
Back to CS103!
Intuiting Nondeterminism

• Nondeterministic machines are a serious departure from physical computers. How can we build up an intuition for them?

• There are two particularly useful frameworks for interpreting nondeterminism:
  • *Perfect positive guessing*
  • *Massive parallelism*
Perfect Positive Guessing

\[ \Sigma \]

Start: \( q_0 \) → \( q_1 \) (a) → \( q_2 \) (b) → \( q_3 \) (a)

Input: \( a \ b \ a \ b \ a \ b \ a \)
Perfect Positive Guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

Input sequence: \[ ababa \]
Perfect Positive Guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

- Start at \( q_0 \)
- \( \Sigma \) transitions
- Part of the word is: \( a b a b a a \)

Diagram:
- Start at \( q_0 \)
- Transitions:
  - \( a \) from \( q_0 \) to \( q_1 \)
  - \( b \) from \( q_1 \) to \( q_2 \)
  - \( a \) from \( q_2 \) to \( q_3 \)
Perfect Positive Guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

Start

\[ a \ b \ a \ b \ a \ a \]
Perfect Positive Guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

\[ \Sigma \]

Start

\[ a \ b \ a \ b \ a \ b \ a \]
Perfect Positive Guessing

\[ q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3 \]

Starting from \( q_0 \), the machine transitions through states \( q_1 \) and \( q_2 \) upon receiving inputs \( a \) and \( b \) respectively, ending at \( q_3 \).

Input sequence: \( \Sigma \)
Perfect Positive Guessing

- We can view nondeterministic machines as having **Magic Superpowers** that enable them to guess choices that lead to an accepting state.
  - If there is at least one choice that leads to an accepting state, the machine will guess it.
  - If there are no choices, the machine guesses any one of the wrong guesses.
- There is no known way to physically model this intuition of nondeterminism – this is quite a departure from reality!
Massive Parallelism

\[
\begin{align*}
q_0 & \xrightarrow{a} q_1 \\
q_1 & \xrightarrow{b} q_2 \\
q_2 & \xrightarrow{a} q_3
\end{align*}
\]

Input symbols: \( \Sigma \)

Initial state: \( q_0 \)

States: \( q_0, q_1, q_2, q_3 \)

Transition arrows:
- \( q_0 \) to \( q_1 \) on input \( a \)
- \( q_1 \) to \( q_2 \) on input \( b \)
- \( q_2 \) to \( q_3 \) on input \( a \)

Loop at \( q_3 \)
Massive Parallelism
Massive Parallelism

\[
\sum_{a, b, a, b, a, b, a, a}
\]
Massive Parallelism

\[ a \quad b \quad a \quad b \quad a \quad b \quad a \]
Massive Parallelism

Start

$q_0$ $q_1$ $q_2$ $q_3$

$\sum$

$a$ $b$ $a$ $b$ $a$ $b$ $a$
Massive Parallelism

\[ \Sigma \]

Start \( q_0 \) \( \rightarrow \) \( a \) \( q_1 \) \( \rightarrow \) \( b \) \( q_2 \) \( \rightarrow \) \( a \) \( q_3 \)

\[ \text{a b a b a b a} \]
Massive Parallelism

We're in at least one accepting state, so there's some path that gets us to an accepting state.

\[\Sigma\]

\[q_0 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{a} q_3\]

\[
\begin{array}{cccccc}
  a & b & a & b & a & a \\
\end{array}
\]
Massive Parallelism

- An NFA can be thought of as a DFA that can be in many states at once.
- At each point in time, when the NFA needs to follow a transition, it tries all the options at the same time.
- (Here's a rigorous explanation about how this works; read this on your own time).
  - Start off in the set of all states formed by taking the start state and including each state that can be reached by zero or more ε-transitions.
  - When you read a symbol $a$ in a set of states $S$:
    - Form the set $S'$ of states that can be reached by following a single $a$ transition from some state in $S$.
    - Your new set of states is the set of states in $S'$, plus the states reachable from $S'$ by following zero or more ε-transitions.
Designing NFAs
Designing NFAs

• **Embrace the nondeterminism!**

• Good model: **Guess-and-check**:  
  - Is there some information that you'd really like to have? Have the machine *nondeterministically guess* that information.
  - Then, have the machine *deterministically check* that the choice was correct.

• The *guess* phase corresponds to trying lots of different options.

• The *check* phase corresponds to filtering out bad guesses or wrong options.
Guess-and-Check

$$L = \{ \ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \ \}$$
Guess-and-Check

\[ L = \{ w \in \{0, 1\}^* \mid w \text{ ends in } 010 \text{ or } 101 \} \]
Guess-and-Check

\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]
Guess-and-Check

\[ L = \{ w \in \{a, b, c\}^* \mid \text{at least one of } a, b, \text{ or } c \text{ is not in } w \} \]

Nondeterministically guess which character is missing.

Deterministically check whether that character is indeed missing.
Just how powerful are NFAs?
Next Time

- **The Powerset Construction**
  - So beautiful. So elegant. So cool!

- **More Closure Properties**
  - Other set-theoretic operations.

- **Language Transformations**
  - What’s the deal with the notation $\Sigma^*$?