Lecture 4:
Non-Regular Languages, Minimizing DFAs
6.045

Announcements:
- Pset 2 is up (as of last night)
  - easier, because fewer lectures!
- How was Pset 1?
DFAs <-> NFAs

Regular Languages <-> Regular Expressions

DEFINITION
Some Languages Are Not Regular:

Limitations on DFAs
Regular or Not?

\[ \Sigma = \{0,1\} \]

\[ C = \{ w \mid w \text{ has equal number of } 1\text{s and } 0\text{s} \} \]

NOT REGULAR!

\[ D = \{ w \mid w \text{ has equal number of occurrences of } 01 \text{ and } 10 \} \]

REGULAR!
$$\Sigma = \{0, 1\}$$

$$D = \{w \mid w \text{ has equal number of occurrences of 01 and 10}\}$$

$$= \{w \mid w = 1, w = 0, \text{ or } w = \varepsilon, \text{ or } w \text{ starts with a 0 and ends with a 0, or } w \text{ starts with a 1 and ends with a 1}\}$$

$$1 + 0 + \varepsilon + 0(0+1)^*0 + 1(0+1)^*1$$

Claim:
A string $w$ has equal occurrences of 01 and 10
$\iff w$ starts and ends with the same bit.
The Pumping Lemma: Structure in Regular Languages

Let L be a regular language

Then there is a positive integer P s.t.

for all strings \( w \in L \) with \( |w| \geq P \)
there are \( x, y, z \) where \( w = xyz \), and:

1. \( |y| > 0 \) (that is, \( y \neq \varepsilon \))
2. \( |xy| \leq P \)
3. For all \( i \geq 0 \), \( xy^iz \in L \)

Why is it called the pumping lemma? The word \( w \) gets pumped into longer and longer strings...
Proof: Let $M$ be a DFA that recognizes $L$

Let $P$ be the number of states in $M$

Let $w$ be a string where $w \in L$ and $|w| \geq P$

We show: $w = xyz$

1. $|y| > 0$
2. $|xy| \leq P$
3. $xy^iz \in L$ for all $i \geq 0$

Claim: There must exist $j$ and $k$ such that $0 \leq j < k \leq P$, and $q_j = q_k$
Let’s prove that \( EQ = \{ w \mid w \text{ has equal number of 1s and 0s} \} \) is not regular.

By contradiction. Assume \( EQ \) is regular. Let \( P \) be as in pumping lemma. Let \( w = 0^P1^P \); note \( w \in EQ \).

If \( EQ \) is regular, then there is some way to write \( w \) as \( w = xyz, |y| > 0, |xy| \leq P \), and for all \( i \geq 0 \), \( xy^iz \) is also in \( EQ \).

Claim: The string \( y \) must be all zeroes.

Why? Because \( |xy| \leq P \) and \( w = xyz = 0^P1^P \).

But then \( xy^3z \) has more 0s than 1s! Contradiction!
Applying the Pumping Lemma

Let's prove that
\[ SQ = \{0^{n^2} \mid n \geq 0\} \] is not regular

Assume \( SQ \) is regular. Let \( w = 0^{P^2} \)

If \( SQ \) is regular, then we can write \( w = xyz, \ |y| > 0, \ |xy| \leq P, \) and for any \( i \geq 0, \ xy^iz \) is also in \( SQ \)

\[ \text{So } xyyz \in SQ. \text{ We have: } xyyz = 0^{P^2 + |y|} \]

Observe that \( 0 < |y| \leq P \)

\[ \text{So } |xyyz| = P^2 + |y| \leq P^2 + P < P^2 + 2P + 1 = (P+1)^2 \]

and \( P^2 < |xyyz| < (P+1)^2 \)

Therefore \( |xyyz| \) is not a perfect square!

Hence \( 0^{P^2 + |y|} = xyyz \notin SQ \), so our assumption must be false.

That is, \( SQ \) is not regular!
Minimizing DFAs
Does this DFA have a minimal number of states?

NO
Is this minimal?

How can we tell in general?
DFA Minimization Theorem:

For every regular language \( A \), there is a unique (up to re-labeling of the states) minimal-state DFA \( M^* \) such that \( A = L(M^*) \).

Furthermore, there is an efficient algorithm which, given any DFA \( M \), will output this unique \( M^* \).

If such algorithms existed for more general models of computation, that would be an engineering breakthrough!!
Note: There isn’t a uniquely minimal NFA
Extending transition function $\delta$ to strings

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$, we extend $\delta$ to a function $\Delta : Q \times \Sigma^* \to Q$ as follows:

$$
\Delta(q, \varepsilon) = q
$$

$$
\Delta(q, \sigma) = \delta(q, \sigma)
$$

$$
\Delta(q, \sigma_1 \ldots \sigma_{k+1}) = \delta(\Delta(q, \sigma_1 \ldots \sigma_k), \sigma_{k+1})
$$

$\Delta(q, w) =$ the state of $M$ reached after reading in $w$, starting from state $q$

Note: $\Delta(q_0, w) \in F \iff M$ accepts $w$

Def. $w \in \Sigma^*$ distinguishes states $q_1$ and $q_2$ iff exactly one of $\Delta(q_1, w), \Delta(q_2, w)$ is a final state
Extending transition function \( \delta \) to strings

Given DFA \( M = (Q, \Sigma, \delta, q_0, F) \), we extend \( \delta \) to a function \( \Delta : Q \times \Sigma^* \rightarrow Q \) as follows:

\[
\begin{align*}
\Delta(q, \varepsilon) &= q \\
\Delta(q, \sigma) &= \delta(q, \sigma) \\
\Delta(q, \sigma_1 \ldots \sigma_{k+1}) &= \delta(\Delta(q, \sigma_1 \ldots \sigma_k), \sigma_{k+1})
\end{align*}
\]

\( \Delta(q, w) = \text{the state of } M \text{ reached after reading in } w, \text{ starting from state } q \)

Note: \( \Delta(q_0, w) \in F \iff M \text{ accepts } w \)

Def. \( w \in \Sigma^* \text{ distinguishes states } q_1 \text{ and } q_2 \) iff
\( \Delta(q_1, w) \in F \iff \Delta(q_2, w) \notin F \)
Distinguishing two states

Def. \( w \in \Sigma^* \textit{ distinguishes} \) states \( q_1 \) and \( q_2 \) iff exactly one of \( \Delta(q_1, w) \), \( \Delta(q_2, w) \) is a final state

I’m in \( q_1 \) or \( q_2 \), but which? How can I tell?

Here... read this
Fix $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ and let $p, q \in Q$

**Definition:**

State $p$ is *distinguishable* from state $q$

iff there is $w \in \Sigma^*$ that distinguishes $p$ and $q$

iff there is $w \in \Sigma^*$ so that exactly one of $\Delta(p, w), \Delta(q, w)$ is a final state

State $p$ is *indistinguishable* from state $q$

iff $p$ is not distinguishable from $q$

iff for all $w \in \Sigma^*$, $\Delta(p, w) \in F \iff \Delta(q, w) \in F$

(EITHER both $\Delta(p, w), \Delta(q, w)$ are in $F$, OR both are not in $F$)

*Pairs of indistinguishable states are redundant...*
Which pairs of states are distinguishable?

Are $q_0$ and $q_1$ distinguishable?

$\varepsilon$ distinguishes all final states from non-final states
The string 10 distinguishes $q_0$ and $q_3$
The string 0 distinguishes $q_1$ and $q_2$
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q, r \in Q$

Define a binary relation $\sim$ on the states of $M$:  
\[ p \sim q \text{ iff } p \text{ is indistinguishable from } q \]
\[ p \nabla q \text{ iff } p \text{ is distinguishable from } q \]

**Proposition:** $\sim$ is an equivalence relation

\[ p \sim p \text{ (reflexive)} \]
\[ p \sim q \Rightarrow q \sim p \text{ (symmetric)} \]
\[ p \sim q \text{ and } q \sim r \Rightarrow p \sim r \text{ (transitive)} \]

**Proof?**
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q, r \in Q$

Proposition: $\sim$ is an **equivalence relation**

Therefore, the relation $\sim$ partitions $Q$ into disjoint equivalence classes

$$[q] := \{ p \mid p \sim q \}$$
Algorithm: MINIMIZE-DFA

Input: DFA $M$

Output: DFA $M_{\text{MIN}}$ such that:

$L(M) = L(M_{\text{MIN}})$

$M_{\text{MIN}}$ has no \textit{inaccessible} states

$M_{\text{MIN}}$ is \textit{irreducible}

\[\]

For all states $p \neq q$ of $M_{\text{MIN}}$, $p$ and $q$ are distinguishable

\[\]

Theorem: $M_{\text{MIN}}$ is the unique minimal DFA that is equivalent to $M$
Intuition:

States of $M_{\text{MIN}}$ = Equivalence classes of states of M

We’ll uncover these equivalent states with a dynamic programming algorithm
The Table-Filling Algorithm

Input: DFA $M = (Q, \Sigma, \delta, q_0, F)$

Output: 
1. $D_M = \{(p, q) | p, q \in Q \text{ and } p \sim q \}$
2. $\text{EQUIV}_M = \{ [q] | q \in Q \}$

Idea:

- We know how to find those pairs of states that the string $\varepsilon$ distinguishes...
- Use this and *iteration* to find those pairs distinguishable with longer strings
- The pairs of states left over will be indistinguishable