CS154

Pumping Lemma,
Minimizing DFAs
Homework 1 is due! (11:59pm tonight...) Homework 2 will appear this afternoon
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 is a way to write \( w = xyz \), where:

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 \textit{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 $i \geq 0$

There must exist $j$ and $k$ such that $0 \leq j < k \leq P$, and $q_j = q_k$
Let’s prove that \( B = \{0^n1^n \mid n \geq 0\} \) is not regular.

By contradiction. Assume \( B \) is regular.

Let \( P \) be the number of states in a DFA for \( B \).

Let \( w = 0^P1^P \)

By the pumping lemma, there is a way to write \( w \) as \( w = xyz \), \(|y| > 0\), \(|xy| \leq P\), and for all \( i \geq 0\), \( xy^iz \) is also in \( B \).

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

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

But then \( xyyz \) has more 0s than 1s \( \text{Contradiction!} \)
Applying the Pumping Lemma

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

is not regular

Assume \( C \) is regular. Let \( w = 0^p1^p \) (\( w \) is in \( C \! \)!) By the pumping lemma, can write \( w = xyz, \ |y| > 0, \ |xy| \leq P \), where for any \( i \geq 0 \), \( xy^iz \) is also in \( C \)

Note that \( |xy| \leq P \), \( w = xyz \), and \( w = 0^p1^p \)

Therefore \( y \) must be all zeroes.

But then \( xyyz \) has more 0s than 1s

**Contradiction!**
Theorem:
\[ B = \{0^{n^2} \mid n \geq 0\} \] is not regular

Assume \( B \) is regular. Let \( w = 0^{p^2} \)

By the pumping lemma, we can write \( w = xyz, \ |y| > 0, \ |xy| \leq P \), and for any \( i \geq 0 \), \( xy^iz \) is also in \( B \)

So we have \( xyyz \in B \). Note that \( xyyz = 0^{p^2+|y|} \)

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

therefore \( P^2 + |y| \leq P^2 + P < P^2 + 2P + 1 = (P+1)^2 \)

\( P^2 < P^2 + |y| < (P+1)^2 \)

therefore \( P^2 + |y| \) is not a perfect square!

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

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

NO
Is this minimal?
Theorem

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

Furthermore, there is an efficient algorithm which, given any DFA $M$, will output this unique $M^*$. 
There isn’t a uniquely minimal NFA
Extending the transition function $\delta$

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 starting in state $q$ and reading in $w$

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

Def. $w \in \Sigma^*$ distinguishes states $q_1$ and $q_2$ iff

$$\Delta(q_1, w) \in F \iff \Delta(q_2, w) \notin F$$
Extending the transition function $\delta$

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 starting in state $q$ and reading in $w$

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
Distinguishing two states

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

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

Here... read this
Distinguishing two states

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.

Ok, I’m accepting! Must have been $q_1$.
Fix \( M = (Q, \Sigma, \delta, q_0, F) \) and let \( p, q \in Q \)

Definition:

State \( p \) is *distinguishable* from state \( q \) iff some \( w \in \Sigma^* \) distinguishes \( p \) and \( q \)

iff there is a string \( w \in \Sigma^* \) 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 \( w \in \Sigma^* \), \( \Delta(p, w) \in F \iff \Delta(q, w) \in F \)

*Pairs of indistinguishable states are redundant... they lead to the same accept/reject behavior!*
\( \varepsilon \) distinguishes accept states and non-accept states
The string 10 distinguishes $q_0$ and $q_3$
The string 0 distinguishes q1 and q2
Fix $M = (Q, \Sigma, \delta, q_0, F)$ and let $p, q, r \in Q$

Define a binary relation $\sim$ on the states of $M$:

$p \sim q$ iff $p$ is indistinguishable from $q$

$p \not\sim q$ iff $p$ is distinguishable from $q$

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

$p \sim p$ (reflexive)

$p \sim q \implies q \sim p$ (symmetric)

$p \sim q$ and $q \sim r \implies p \sim r$ (transitive)

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

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

Proposition: $\sim$ is an equivalence relation

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

Input: DFA M

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

- $L(M) = L(M_{\text{MIN}})$
- $M_{\text{MIN}}$ has no inaccessible states
- $M_{\text{MIN}}$ is 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 $M_{MIN}$ will be the *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. $D_M = \{ (p, q) \mid p, q \in Q \text{ and } p \not\sim q \}$
2. $\text{EQUIV}_M = \{ [q] \mid q \in Q \}$

High-Level Idea:

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

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

Output:
1. $D_M = \{ (p, q) \mid p, q \in Q \text{ and } p \not\sim q \}$
2. $\text{EQUIV}_M = \{ [q] \mid q \in Q \}$

Base Case: For all $(p, q)$ such that $p$ accepts and $q$ rejects $\Rightarrow p \not\sim q$
The Table-Filling Algorithm

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

Output:  
1. $D_M = \{ (p, q) \mid p, q \in Q \text{ and } p \not\sim q \}$
2. $\text{EQUIV}_M = \{ [q] \mid q \in Q \}$

Base Case: For all $(p, q)$ such that $p$ accepts and $q$ rejects $\Rightarrow p \not\sim q$

Iterate: If there are states $p$, $q$ and symbol $\sigma \in \Sigma$ satisfying:

\[ \delta(p, \sigma) = p' \quad \text{Mark} \]
\[ \delta(q, \sigma) = q' \]

\[ p \not\sim q \]

Repeat until no more $D$’s can be added.
Claim: If \((p, q)\) is marked \(D\) by the Table-Filling algorithm, then \(p \not\sim q\)

Proof: By induction on the length of the string distinguishing \(p\) and \(q\).

If \((p, q)\) is marked \(D\) at the start, then one’s in \(F\) and one isn’t, so \(\varepsilon\) distinguishes \(p\) and \(q\).

Suppose \((p, q)\) is marked \(D\) at a later point.

Then there are states \(p', q'\) such that:

1. \((p', q')\) are marked \(D\) \(\Rightarrow p' \not\sim q'\) (by induction)

\(\Rightarrow\) There is a string \(w\) s.t. \(\Delta(p', w) \in F\) \(\Leftrightarrow \Delta(q', w) \notin F\)

2. \(p' = \delta(p, \sigma)\) and \(q' = \delta(q, \sigma)\), where \(\sigma \in \Sigma\)

The string \(\sigma w\) distinguishes \(p\) and \(q\)!
Claim: If \((p, q)\) is not marked \(D\) by the Table-Filling algorithm, then \(p \sim q\)

Proof (by contradiction):
Suppose the pair \((p, q)\) is not marked \(D\) by the algorithm, yet \(p \not\sim q\) (call this a “bad pair”)

Then there is a string \(w\) such that \(|w| > 0\) and:
\[
\Delta(p, w) \in F \quad \text{and} \quad \Delta(q, w) \notin F
\]

(Why is \(|w| > 0\)?)

Of all such bad pairs, let \(p, q\) be a pair with the \emph{shortest} distinguishing string \(w\)
Claim: If \((p, q)\) is not marked \(D\) by the Table-Filling algorithm, then \(p \sim q\)

Proof (by contradiction):
Suppose the pair \((p, q)\) is not marked \(D\) by the algorithm, yet \(p \not\sim q\) (call this a “bad pair”)

Of all such bad pairs, let \(p, q\) be a pair with the shortest distinguishing string \(w\)

\[ \Delta(p, w) \in F \text{ and } \Delta(q, w) \not\in F \quad (\text{Why is } |w| > 0?) \]

We have \(w = \sigma w'\), for some string \(w'\) and some \(\sigma \in \Sigma\)

Let \(p' = \delta(p, \sigma)\) and \(q' = \delta(q, \sigma)\)

Then \((p', q')\) is also a bad pair, but with a SHORTER \(w'\)!