15-453
FORMAL LANGUAGES,
AUTOMATA AND
COMPUTABILITY
THEOREM

For every regular language $L$, there exists a UNIQUE (up to re-labeling of the states) minimal DFA $M_{\text{min}}$ such that $L = L(M_{\text{min}})$

Minimal means wrt number of states
PROOF

1. Let $M$ be a DFA for $L$ (wlog, assume no inaccessible states)
2. For pairs of states $(p,q)$ define:
   $p$ distinguishable from $q$ and $p$ indistinguishable from $q$ ($p \sim q$).
3. Table-filling algorithm: first distinguish final from non-final states and then work backwards to distinguish more pairs.
4. What’s left over are exactly the indistinguishable pairs, ie $\sim$ related pairs. Needs proof.
5. $\sim$ is an equivalence relation so partitions the states into equivalence classes, $E_M$

6. Define $M_{\text{min}}$

Define: $M_{\text{MIN}} = (Q_{\text{MIN}}, \Sigma, \delta_{\text{MIN}}, q_{0\text{ MIN}}, F_{\text{MIN}})$

$Q_{\text{MIN}} = E_M$, $q_{0\text{ MIN}} = [q_0]$, $F_{\text{MIN}} = \{[q] \mid q \in F\}$

$\delta_{\text{MIN}}( [q], \sigma ) = [ \delta( q, \sigma ) ]$ show well defined

Claim: $\hat{\delta}_{\text{MIN}}( [q], w ) = [ \hat{\delta}( q, w ) ]$, $w \in \Sigma^*$

So: $\hat{\delta}_{\text{MIN}}( [q_0], w ) = [ \hat{\delta}( q_0, w ) ]$, $w \in \Sigma^*$

Follows: $M_{\text{MIN}} \equiv M$
But is $M_{\text{min}}$ unique minimum?

Yes, because if $M' \equiv M$ and minimum then $M'$ has no inaccessible states and is irreducible and...

Theorem. $M_{\text{min}}$ is isomorphic to any $M'$ with the above properties (need to give mapping and prove it has all the needed properties: everywhere defined, well defined, 1-1, onto, preserves transitions, and \{final states\} map onto \{final states\})

So $M_{\text{min}}$ is isomorphic to \textit{any} minimum $M' \equiv M$
How can we prove that two DFAs are equivalent?

One way: Minimize

Another way: Let $C = (\neg A \cap B) \cup (A \cap \neg B)$
Then, $A = B \iff C = \emptyset$

$C$ is the “disjoint union”
CONTEXT-FREE GRAMMARS
AND PUSH-DOWN AUTOMATA
TUESDAY Jan 28
NONE OF THESE ARE REGULAR

$\Sigma = \{0, 1\}, \quad L = \{ 0^n1^n \mid n \geq 0 \}$

$\Sigma = \{a, b, c, \ldots, z\}, \quad L = \{ w \mid w = w^R \}$

$\Sigma = \{ (, ) \}, \quad L = \{ \text{balanced strings of parens} \}$

(, ()(), ((()()) are in $L$, (, ()), (()()) are not in $L$

PUSHDOWN AUTOMATA (PDA)

FINITE STATE CONTROL

INPUT

STACK (Last in, first out)
A brief history of the stack, Sten Henriksson, Computer Science Department, Lund University, Lund, Sweden.
Non-deterministic
PDA that recognizes $L = \{ 0^n1^n \mid n \geq 0 \}$
**Definition:** A *(non-deterministic)* PDA is a 6-tuple $P = (Q, \Sigma, \Gamma, \delta, q_0, F)$, where:

- **$Q$** is a finite set of states
- **$\Sigma$** is the input alphabet
- **$\Gamma$** is the stack alphabet
- **$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \rightarrow 2^{Q \times \Gamma_\varepsilon}$** is the set of subsets of $Q \times \Gamma_\varepsilon$
- **$q_0 \in Q$** is the start state
- **$F \subseteq Q$** is the set of accept states

$
\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}, \quad \Gamma_\varepsilon = \Gamma \cup \{\varepsilon\}$

**Push** and **Pop** operations also apply to stack transitions.
Let $w \in \Sigma^*$ and suppose $w$ can be written as $w_1 \ldots w_n$ where $w_i \in \Sigma_\epsilon$ (recall $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$).

Then $P$ accepts $w$ if there are $r_0, r_1, \ldots, r_n \in Q$ and $s_0, s_1, \ldots, s_n \in \Gamma^*$ (sequence of stacks) such that

1. $r_0 = q_0$ and $s_0 = \epsilon$ ($P$ starts in $q_0$ with empty stack)

2. For $i = 0, \ldots, n-1$:
   $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, where $s_i = at$ and $s_{i+1} = bt$ for some $a, b \in \Gamma_\epsilon$ and $t \in \Gamma^*$ ($P$ moves correctly according to state, stack and symbol read)

3. $r_n \in F$ ($P$ is in an accept state at the end of its input)
\[
Q = \{q_0, q_1, q_2, q_3\} \quad \Sigma = \{0, 1\} \quad \Gamma = \{\$, 0, 1\}
\]

\[
\delta : Q \times \Sigma \times \Gamma \rightarrow 2^{Q \times \Gamma}
\]

\[
\delta(q_1, 1, 0) = \{ (q_2, \varepsilon) \} \quad \delta(q_2, 1, 1) = \emptyset
\]
EVEN-LENGTH PALINDROMES

$\Sigma = \{a, b, c, \ldots, z\}$

Diagram:

- $q_0 \xrightarrow{\varepsilon, \varepsilon} q_1$
- $q_1 \xrightarrow{\sigma, \varepsilon} \sigma$
- $q_1 \xrightarrow{\varepsilon, \varepsilon} \varepsilon$
- $q_2 \xrightarrow{\varepsilon, \$} \varepsilon$
- $q_2 \xrightarrow{\sigma, \sigma} \varepsilon$
- $q_3 \xrightarrow{\varepsilon, \$} \varepsilon$

Examples:
- zeus sees suez
- Madamimadam

(How to recognize odd-length palindromes?)
Build a PDA to recognize
\[ L = \{ a^i b^j c^k | i, j, k \geq 0 \text{ and } (i = j \text{ or } i = k) \} \]
Build a PDA to recognize
\[ L = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } (i = j \text{ or } i = k) \} \]
Build a PDA to recognize
$L = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } (i = j \text{ or } i = k) \}$
CONTEXT-FREE GRAMMARS

“Colorless green ideas sleep furiously.”

Noam Chomsky (1957)
Non-deterministic

We say: \( 00\#11 \) is generated by the Grammar

Derivation:

\[ A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11 \]

(yields)

A \( \Rightarrow^* \) 00\#11
CONTEXT-FREE GRAMMARS

A → 0A1
A → B
B → #

A ⇒ 0A1 ⇒ 00A11 ⇒ 00B11 ⇒ 00#11

(yields)

A ⇒* 00#11
(derives)

We say: 00#11 is generated by the Grammar

Deterministic CFGs??
CONTEXT-FREE GRAMMARS

A → 0A1
A → B
B → #

A → 0A1 | B
B → #
SNOOP'S GRAMMAR
(courtesy of Luis von Ahn)

<PHRASE> → <FILLER><PHRASE>
<PHRASE> → <START WORD><END WORD> DUDE
<FILLER> → LIKE
<FILLER> → UMM
<START WORD> → FO
<START WORD> → FA
<END WORD> → SHO
<END WORD> → SHAZZY
<END WORD> → SHEEZY
<END WORD> → SHIZZLE
SNOOP’S GRAMMAR
(courtesy of Luis von Ahn)

Generate:
Umm Like Umm Umm Fa Shizzle Dude
Fa Sho Dude
A context-free grammar (CFG) is a tuple $G = (V, \Sigma, R, S)$, where:

- $V$ is a finite set of variables
- $\Sigma$ is a finite set of terminals (disjoint from $V$)
- $R$ is set of production rules of the form $A \rightarrow W$, where $A \in V$ and $W \in (V \cup \Sigma)^*$
- $S \in V$ is the start variable
A context-free grammar (CFG) is a tuple $G = (V, \Sigma, R, S)$, where:

- $V$ is a finite set of variables
- $\Sigma$ is a finite set of terminals (disjoint from $V$)
- $R$ is set of production rules of the form $A \rightarrow W$, where $A \in V$ and $W \in (V \cup \Sigma)^*$
- $S \in V$ is the start variable

$L(G) = \{ w \in \Sigma^* \mid S \Rightarrow^* w \}$ Strings Generated by $G$

A Language $L$ is context-free if there is a CFG that generates precisely the strings in $L$
A context-free grammar (CFG) is a tuple $G = (V, \Sigma, R, S)$, where:

- $V$ is a finite set of variables
- $\Sigma$ is a finite set of terminals (disjoint from $V$)
- $R$ is set of production rules of the form $A \rightarrow W$, where $A \in V$ and $W \in (V \cup \Sigma)^*$
- $S \in V$ is the start variable

$G = \{ \{ S \}, \{ 0, 1 \}, R, S \}$

$R = \{ S \rightarrow 0S1, S \rightarrow \epsilon \}$

$L(G) =$
A context-free grammar (CFG) is a tuple 
\( G = (V, \Sigma, R, S) \), where:

- \( V \) is a finite set of variables
- \( \Sigma \) is a finite set of terminals (disjoint from \( V \))
- \( R \) is set of production rules of the form \( A \rightarrow W \), where \( A \in V \) and \( W \in (V \cup \Sigma)^* \)
- \( S \in V \) is the start variable

\[ G = \{ \{S\}, \{0,1\}, R, S \} \quad R = \{ S \rightarrow 0S1, S \rightarrow \epsilon \} \]

\[ L(G) = \{ 0^n1^n \mid n \geq 0 \} \quad \text{Strings Generated by } G \]
WRITE A CFG FOR EVEN-LENGTH PALINDROMES

\[ S \rightarrow \sigma S \sigma \text{ for all } \sigma \in \Sigma \]
\[ S \rightarrow \varepsilon \]
WRITE A CFG FOR THE EMPTY SET

G = { {S}, Σ, ∅, S }
PARSE TREES

A → 0A1
A → B
B → #

A 0A1 00A11 00B11 00#11
Build a parse tree for \(a + a \times a\)
Definition: a string is derived **ambiguously** in a context-free grammar if it has more than one parse tree.

Definition: a grammar is **ambiguous** if it generates some string ambiguously.

See $G_4$ for unambiguous standard arithmetic precedence [adds parens (,)]

$L = \{ a^i b^j c^k | i, j, k \geq 0 \text{ and } (i = j \text{ or } j = k) \}$ is *inherently ambiguous* (xtra credit)

**Undecidable** to tell if a language has unambiguous parse trees (Post’s problem)
NOT REGULAR

\[ \Sigma = \{0, 1\}, \quad L = \{ 0^n1^n \mid n \geq 0 \} \]

But \( L \) is CONTEXT FREE

\[
\begin{align*}
A & \rightarrow 0A1 \\
A & \rightarrow \varepsilon
\end{align*}
\]

WHAT ABOUT?

\[ \Sigma = \{0, 1\}, \quad L_1 = \{ 0^n1^n 0^m \mid m, n \geq 0 \} \]
\[ \Sigma = \{0, 1\}, \quad L_2 = \{ 0^n1^m 0^n \mid m, n \geq 0 \} \]
\[ \Sigma = \{0, 1\}, \quad L_3 = \{ 0^m1^n 0^n \mid m=n \geq 0 \} \]
WHAT ABOUT?

\[ \Sigma = \{0, 1\}, \quad L_1 = \{ 0^n1^n 0^m \mid m, n \geq 0 \} \]

\[ \Sigma = \{0, 1\}, \quad L_2 = \{ 0^n1^m 0^n \mid m, n \geq 0 \} \]

\[ \Sigma = \{0, 1\}, \quad L_3 = \{ 0^m1^n 0^n \mid m=n \geq 0 \} \]
WHAT ABOUT?

$\Sigma = \{0, 1\}, \ L_1 = \{ \ 0^n1^n \ 0^m | \ m, n \geq 0 \ \}$
S $\rightarrow$ AB
A $\rightarrow$ 0A1 $|$ $\epsilon$
B $\rightarrow$ 0B $|$ $\epsilon$

$\Sigma = \{0, 1\}, \ L_2 = \{ \ 0^n1^m \ 0^n | \ m, n \geq 0 \ \}$
S $\rightarrow$ 0S0 $|$ A
A $\rightarrow$ 1A $|$ $\epsilon$

$\Sigma = \{0, 1\}, \ L_3 = \{ \ 0^m1^n \ 0^n | \ m=n \geq 0 \ \}$
THE PUMPING LEMMA FOR CFGs

Let $L$ be a context-free language

Then there is a $P$ such that

if $w \in L$ and $|w| \geq P$

then can write $w = uv^ixyz$, where:

1. $|vy| > 0$

2. $|vxy| \leq P$

3. For every $i \geq 0$, $uv^ixyz \in L$
WHAT ABOUT?

\[ \Sigma = \{0, 1\}, \quad L_3 = \{0^m1^n0^n | m=n \geq 0\} \]

Choose \(w = 0^P 1^P 0^P\).

By the **Pumping Lemma**, we can write \(w = uvxyz\) with \(|vy| > 0, |vxy| \leq P\) such that pumping \(v\) together with \(y\) will produce another word in \(L_3\).

Since \(|vxy| \leq P\), \(vxy = 0^a1^b\), or \(vxy = 1^a0^b\).
WHAT ABOUT?

\[ \Sigma = \{0, 1\}, \quad L_3 = \{0^m1^n0^n \mid m=n \geq 0\} \]

Choose \( w = 0^P1^P0^P \).

By the **Pumping Lemma**, we can write \( w = uvxyz \) with \(|vy| > 0, |vxy| \leq P\) such that pumping \( v \) together with \( y \) will produce another word in \( L_3 \).

Since \(|vxy| \leq P\), \( vxy = 0^a1^b \), or \( vxy = 1^a0^b \).

Pumping in the first case will unbalance with the 0’s at the end; in the second case, will unbalance with the 0’s at the beginning. **Contradiction.**
THE PUMPING LEMMA FOR CFGs

Let $L$ be a context-free language

Then there is a $P$ such that
if $w \in L$ and $|w| \geq P$

then can write $w = uv^ixy^iz$, where:

1. $|vy| > 0$

2. $|vxy| \leq P$

3. For every $i \geq 0$, $uv^ixy^iz \in L$
Idea of Proof: If \( w \) is long enough, then any parse tree for \( w \) must have a path that contains a variable more than once.
Formal Proof:

Let $b$ be the maximum number of symbols (length) on the right-hand side of any rule.

If the height of a parse tree is $h$, the length of the string generated by that tree is at most: $b^h$

Let $|V|$ be the number of variables in $G$.

Define $P = b^{|V|+1}$.

Let $w$ be a string of length at least $P$.

Let $T$ be a parse tree for $w$ with a minimum number of nodes.

$b^{|V|+1} = P \leq |w| \leq b^h$.

$T$ must have height $h$ at least $|V|+1$. 
Let $T$ be a parse tree for $w$ with a minimum number of nodes. $T$ must have height at least $|V| + 1$.

The longest path in $T$ must have $\geq |V| + 1$ variables.

Select $R$ to be a variable that repeats among the lowest $|V| + 1$ variables (in the path).

1. $|vy| > 0$
2. $|vxy| \leq 1$

Let $T$ be a parse tree for $w$ with a minimum number of nodes. $T$ must have height $|V| + 1$.
The longest path in $T$ must have $\geq |V|+1$ variables.

Select $R$ to be a variable in $T$ that repeats, among the lowest $|V|+1$ variables in the tree.

1. $|vy| > 0$ since $T$ has minimum number of nodes.
2. $|vxy| \leq P$ since $|vxy| \leq b^{|V|+1} = P$. 

---

Diagram showing the structure of $T$ with variables $u$, $v$, $x$, $y$, $z$, and $R$. The diagram illustrates the path and the nodes $u$, $v$, $x$, $y$, and $z$. The variable $R$ is selected as the one that repeats within the lowest $|V|+1$ variables.