15-453

FORMAL LANGUAGES, AUTOMATA AND COMPUTABILITY

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, $A = B \Leftrightarrow C = \varnothing$

CONTEXT-FREE GRAMMARS AND PUSH-DOWN AUTOMATA

TUESDAY Jan 28

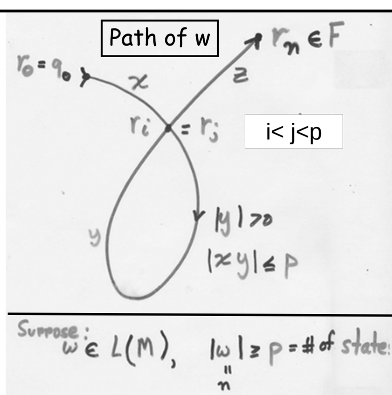
NONE OF THESE ARE REGULAR

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

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

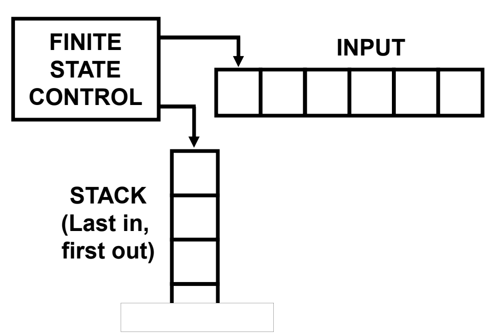
Σ = { (,) }, L = { balanced strings of parens }

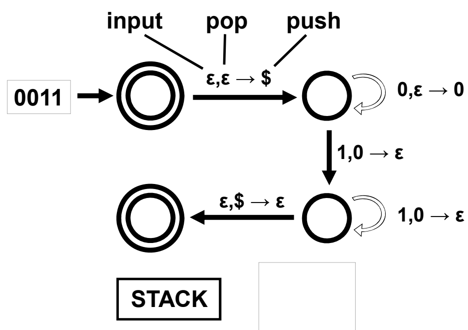
(), ()(), (()()) **are in L, (, ()), ())(() are no**t in L



The Pumping Lemma (for Regular Languages)

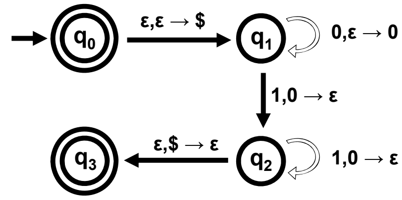
PUSHDOWN AUTOMATA (PDA)





Non-deterministic

PDA that recognizes $L = \{ 0^n1^n \mid n \geq 0 \}$



$Q = \{q_0, q_1, q_2, q_3\}$ $\Sigma = \{0,1\}$ $\Gamma = \{\$,0,1\}$

$\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to 2^{Q \times \Gamma_\varepsilon}$

$\delta(q_1,1,0) = \{ (q_2,\varepsilon) \}$ $\delta(q_2,1,1) = \varnothing$

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**
- **Σ is the input alphabet**
- **Γ is the stack alphabet**
- $\delta : Q \times \Sigma_\varepsilon \times \Gamma_\varepsilon \to 2^{Q \times \Gamma_\varepsilon}$ (pop, push)
- **$q_0 \in Q$ is the start state**
- **$F \subseteq Q$ is the set of accept states**

$2^{Q \times \Gamma_\varepsilon}$ is the set of subsets of $Q \times \Gamma_\varepsilon$

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

Let $w \in \Sigma^*$ and suppose w can be written as $w_1 \ldots w_n$ where $w_i \in \Sigma_\varepsilon$ (recall $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$)

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**

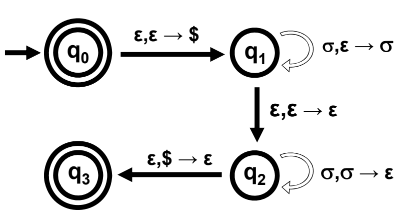
1. $r_0 = q_0$ and $s_0 = \varepsilon$ (**P** starts in **q_0** with empty stack)

2. For i = 0, ..., 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 $a, b \in \Gamma_\varepsilon$ 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)

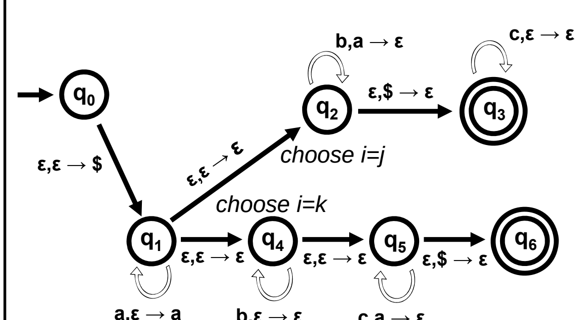
EVEN-LENGTH PALINDROMES

Σ = {a, b, c, …, z}



zeus sees suez
Madamimadam
(PDA to recognize odd length palindromes?)

Build a PDA to recognize
$L = \{ a^ib^jc^k \mid i, j, k \geq 0 \text{ and } (i = j \text{ or } i = k) \}$



CONTEXT-FREE GRAMMARS

"Colorless green ideas sleep *furiously*."

CONTEXT-FREE GRAMMARS

A → 0A1
A → B
B → #

A → 0A1 | B
B → #

CONTEXT-FREE GRAMMARS

start variable

production rules

A → 0A1
A → B
B → #

variables

terminals

A ⇒ 0A1 ⇒ 00A11 ⇒ 00B11 ⇒ 00#11

⇒(yields)

Derivation

A ⇒* 00#11

(derives)

Non-deterministic

We say: **00#11** is *generated* by the Grammar

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

CONTEXT-FREE GRAMMARS

A context-free grammar (CFG) is a tuple G = (V, Σ, R, S), where:

V is a finite set of variables

Σ is a finite set of terminals (disjoint from V)

R is set of production rules of the form A → W, where A ∈ V and W ∈ (V∪Σ)*

S ∈ V is the start variable

CONTEXT-FREE LANGUAGES

A context-free grammar (CFG) is a tuple G = (V, Σ, R, S), where:

V is a finite set of variables

Σ is a finite set of terminals (disjoint from V)

R is set of production rules of the form A → W, where A ∈ V and W ∈ (V∪Σ)*

S ∈ V is the start variable

L(G) = {w ∈ Σ* | S ⇒* w} Strings Generated by G

A Language L is context-free if there is a CFG that generates precisely the string in L

CONTEXT-FREE LANGUAGES

A context-free grammar (CFG) is a tuple G = (V, Σ, R, S), where:

V is a finite set of variables

Σ 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 = { $S \rightarrow 0S1$, $S \rightarrow \varepsilon$ }

L(G) =

CONTEXT-FREE LANGUAGES

A context-free grammar (CFG) is a tuple G = (V, Σ, R, S), where:

V is a finite set of variables

Σ 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 = { $S \rightarrow 0S1$, $S \rightarrow \varepsilon$ }

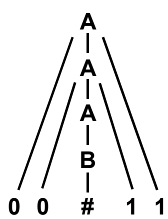
L(G) = { 0^n1^n | $n \geq 0$ } Strings Generated by G

WRITE A CFG FOR EVEN-LENGTH PALINDROMES

WRITE A CFG FOR THE EMPTY SET

PARSE TREES

$A \rightarrow 0A1$
$A \rightarrow B$
$B \rightarrow \#$



$A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 00B11 \Rightarrow 00\#11$

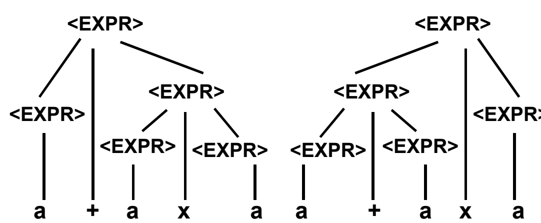
<EXPR> → <EXPR> + <EXPR>

<EXPR> → <EXPR> x <EXPR>

<EXPR> → (<EXPR>)

<EXPR> → a

Build a parse tree for a + a x 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

NOT REGULAR

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

But L is CONTEXT FREE

$A \rightarrow 0A1$
$A \rightarrow \varepsilon$

WHAT ABOUT?

$\Sigma = \{0, 1\}$, $L_1 = \{ 0^n1^n 0^m \mid m,n \geq 0 \}$

$\Sigma = \{0, 1\}$, $L_2 = \{ 0^n1^m 0^n \mid m,n \geq 0 \}$

$\Sigma = \{0, 1\}$, $L_3 = \{ 0^m1^n 0^n \mid m=n \geq 0 \}$

WHAT ABOUT?

$\Sigma = \{0, 1\}$, $L_1 = \{ 0^n1^n 0^m \mid m, n \geq 0 \}$

$\Sigma = \{0, 1\}$, $L_2 = \{ 0^n1^m 0^n \mid m, n \geq 0 \}$

$\Sigma = \{0, 1\}$, $L_3 = \{ 0^m1^n 0^n \mid m=n \geq 0 \}$

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 = uvxyz, where:

1. **$|vy| > 0$**
2. **$|vxy| \leq P$**
3. **For *every* $i \geq 0$, $uv^ixy^iz \in L$**

WHAT ABOUT?

$\Sigma = \{0, 1\}$, $L_3 = \{ 0^m1^n 0^n \mid m=n \geq 0 \}$

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 $\mathbf{L_3}$
Since $|vxy| \leq P$, $vxy = 0^a1^b$, or $vxy = 1^a 0^b$.

WHAT ABOUT?

$\Sigma = \{0, 1\}$, $L_3 = \{ 0^m1^n 0^n \mid m=n \geq 0 \}$

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 $\mathbf{L_3}$
Since $|vxy| \leq P$, $vxy = 0^a1^b$, or $vxy = 1^a 0^b$.

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

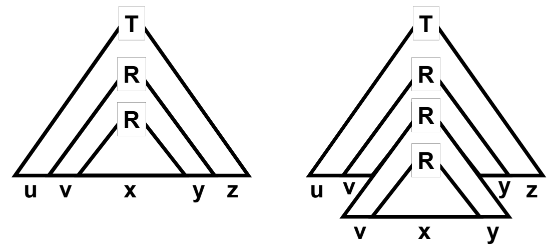
Let L be a context-free language

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

then can write w = uvxyz, 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.

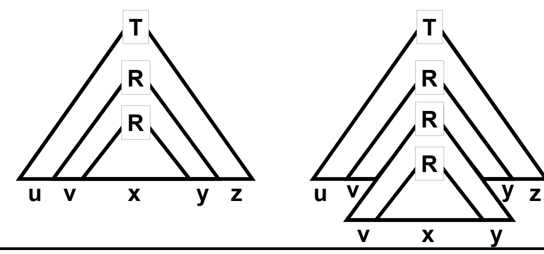
$b^{|V|+1} = P \leq |w| \leq b^h$

T must have height h at least |V|+1

The longest path in T must have ≥ |V|+1 variable

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 minimun # nodes**
2. **$|vxy| \leq P$ since $|vxy| \leq b^{|V|+1} = P$**



EQUIVALENCE OF CFGs and PDAs

A Language L is generated by a CFG
⇔
L is recognized by a PDA

WWW.FLAC.WS

Read the rest of Chapter 2 for next time