Lecture 9

HW2 out! (Due Monday Oct 16^{th})

Today: Finish Pushdown automata (PDAs)
Context Free grammars

PDA Example 1

"$a, b \to c$" means when reading input symbol a, if b is symbol on top of stack, replace b by c

"$a, b \to \varepsilon$" means if reading input symbol a, can pop b off stack

"$a, \varepsilon \to c$" means if reading symbol a, push c onto top of stack

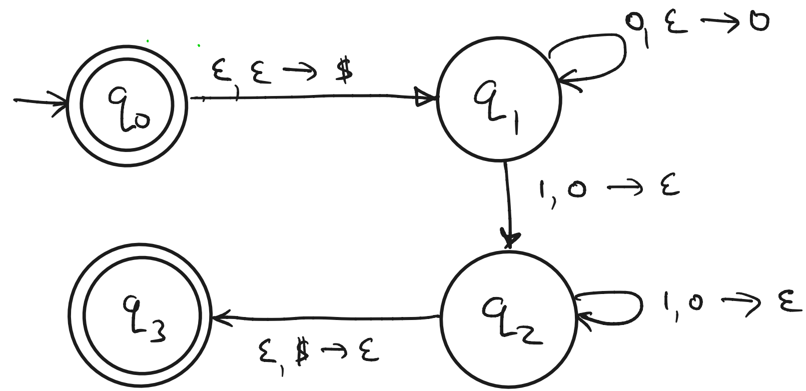
$Q = \{q_0, q_1, q_2, q_3\}$

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

$\Gamma = \{0, \$\}$

$F = \{q_0, q_3\}$

q_0



PDA accepts an input w if there exists a computation path starting in q_0 and ending in an accept state

PDA (Formal Description)

A PDA is described by a 6-tuple $M = (\underset{\text{states}}{Q}, \underset{\text{input alphabet}}{\Sigma}, \underset{\text{stack alphabet}}{\Gamma}, \delta, \underset{\text{start state}}{q_0}, \underset{\text{accept states}}{F})$

$$\delta : Q \times \{\Sigma \cup \varepsilon\} \times \{\Gamma \cup \varepsilon\} \to \mathcal{P}(Q \times \{\Gamma \cup \varepsilon\})$$

M <u>accepts</u> w if w can be written as $w = w_1 w_2 w_3 \ldots w_m$, where each $w_i \in \{\Sigma \cup \varepsilon\}$, and $\exists$ a sequence of states $r_0, r_1, \ldots, r_m \in Q$ and $\exists$ sequence of strings $\underbrace{s_0, s_1, \ldots, s_m}_{s_i = \text{contents of stack at time } i} \in \Gamma^*$ satisfying:

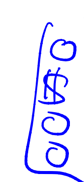
① $r_0 = q_0$, $s_0 = \varepsilon$ (start state is q_0, stack initially empty)

② for all $i = 0, 1, \ldots, m-1$ $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ where $s_i = at$, $s_{i+1} = bt$ $\quad a, b \in \Gamma \cup \varepsilon$, $t \in \Gamma^*$

(M moves according to transition function δ)

③ $r_m \in F$ (final state is an accept state)

$s_i = 0\$00$



PDA (Formal Description)

A PDA is described by a 6-tuple $M = (\underbrace{Q}_{\text{states}}, \underbrace{\Sigma}_{\text{input alphabet}}, \underbrace{\Gamma}_{\text{tape alphabet}}, \delta, \underbrace{q_0}_{\text{start state}}, \underbrace{F}_{\text{accept states}})$

$$\delta : Q \times \{\Sigma \cup \varepsilon\} \times \{\Gamma \cup \varepsilon\} \to P(Q \times \{\Gamma \cup \varepsilon\})$$

M <u>accepts</u> w if w can be written as $w = w_1 w_2 w_3 \ldots w_m$, where each $w_i \in \{\Sigma \cup \varepsilon\}$, and $\exists$ a sequence of states $r_0, r_1, \ldots, r_m \in Q$ and $\exists$ sequence of strings $\underbrace{s_0, s_1, \ldots, s_m}_{s_i = \text{contents of stack at time } i} \in \Gamma^*$ satisfying:

① $r_0 = q_0$, $s_0 = \varepsilon$ (start state is q_0, stack initially empty)

② for all $i = 0, 1, \ldots, m-1$ $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$ where $s_i = at$, $s_{i+1} = bt$, $a, b \in \Gamma \cup \varepsilon$, $t \in \Gamma^*$

(M moves according to transition function δ)

③ $r_m \in F$ (final state is an accept state)

$$\mathcal{L}(M) = \{ w \in \Sigma^* \mid M \text{ accepts } w \}$$

A language is a CFL if some PDA accepts it

PDA (Formal Description)

A PDA is described by a 6-tuple $M = (\underbrace{Q}_{\text{states}}, \underbrace{\Sigma}_{\text{input alphabet}}, \underbrace{\Gamma}_{\text{tape alphabet}}, \delta, \underbrace{q_0}_{\text{start state}}, \underbrace{F}_{\text{accept states}})$

$$\delta : Q \times \{\Sigma \cup \varepsilon\} \times \{\Gamma \cup \varepsilon\} \rightarrow P(Q \times \{\Gamma \cup \varepsilon\})$$

<u>Notes</u>: We only accept if we are in an accept state when all of w is processed.

Note that we can accept a string w even if stack is not empty at end of processing w.

The CFL's include all regular languages

But there are languages that are CFL's that aren't regular. <u>Example</u>: $L = \{0^n 1^n \mid n \geq 0\}$

Example 2

$L = \{ ww^R \mid w \in \{0,1\}^* \}$

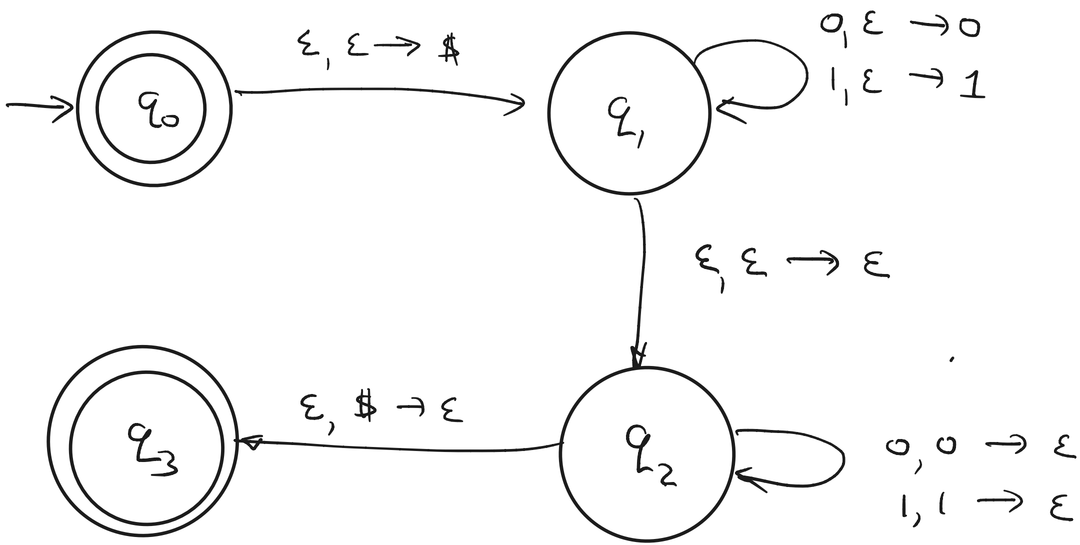
$\left(L \neq \{ w \mid w = w^R \} \right.$

Idea

- Start in q_0: push \$ onto stack, + go to state q_1
- Read symbols + push them onto stack
- at each point, we can nondeterministically guess we're at middle of string (by changing to state q_2)
- when in q_2 read next input symbol + check if it matches top symbol on stack + if so pop top symbol off stack
- guess end of string + if we see "\$" on top of stack go to q_3 (accept state)

Example 2

$L = \{ ww^R \mid w \in \{0,1\}^* \}$

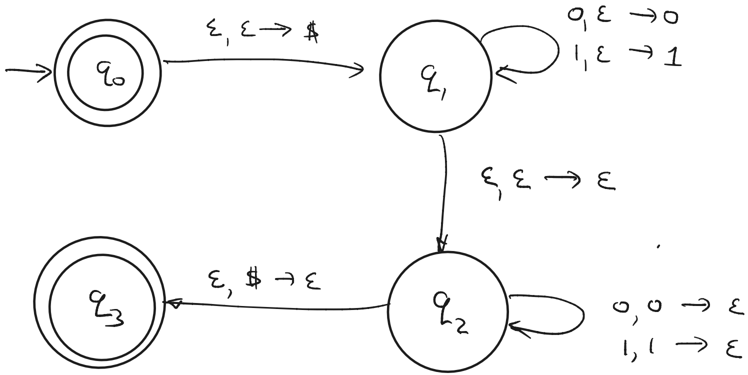


$\delta(q_0, \varepsilon, \varepsilon) = \{(q_1, \$)\}$

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

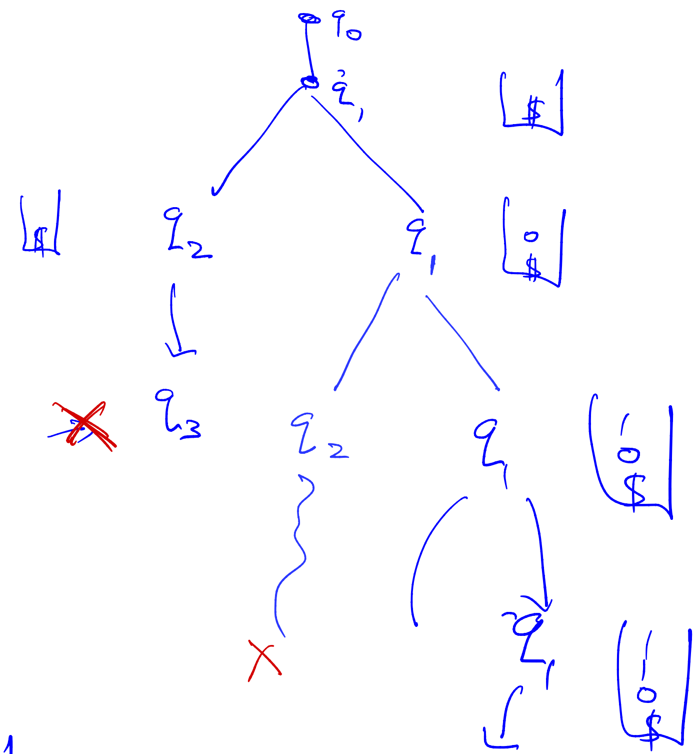
$\delta(q, 1, \varepsilon) \to \{(q_1, 1)\}$

$\delta(q_1, \varepsilon, \varepsilon) \to \{(q_2, \varepsilon)\}$

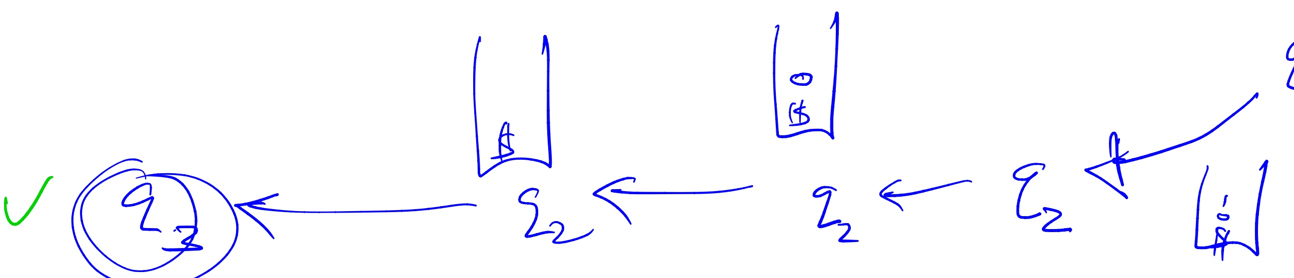


q_0
$\varepsilon, \varepsilon \to \$$
q_1
$0, \varepsilon \to 0$
$1, \varepsilon \to 1$
$\varepsilon, \varepsilon \to \varepsilon$
q_3
$\varepsilon, \$ \to \varepsilon$
q_2
$0, 0 \to \varepsilon$
$1, 1 \to \varepsilon$

$w = 011110$



q_0
q_1
$\$$
q_2
q_1
0
$\$$
q_3
q_2
q_1
1
0
$\$$
q_1
1
1
0
$\$$
q_2



q_3
q_2
$\$$
q_2
0
$\$$
q_2
1
0
$\$$

Example 3.

Any Regular Language is accepted by some PDA.

Let L be regular, and let $M = (Q, \Sigma, F, q_0, \delta)$ be a DFA accepting L.

Corresponding PDA for L: $N = (Q, \Sigma, \Gamma, F, q_0, \delta')$

where $\Gamma = \phi$

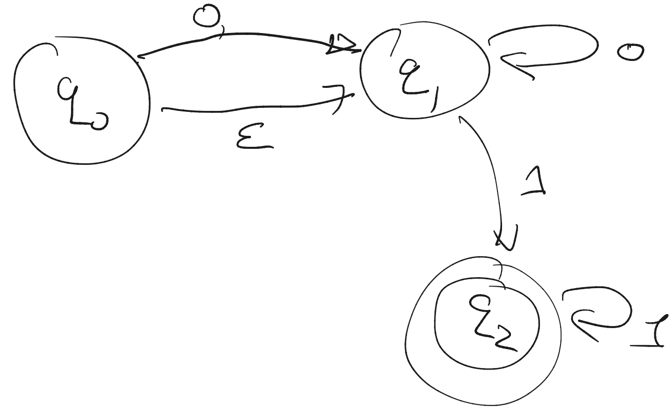
δ' : $(q, a, \varepsilon) = \{(q', \varepsilon)\}$

$$\delta(a, q) = (q') \xRightarrow{\text{becomes}} \delta'(a, \varepsilon, q) = \{(\varepsilon, q')\}$$

(DFAs/NFAs are PDA where stack is always empty)

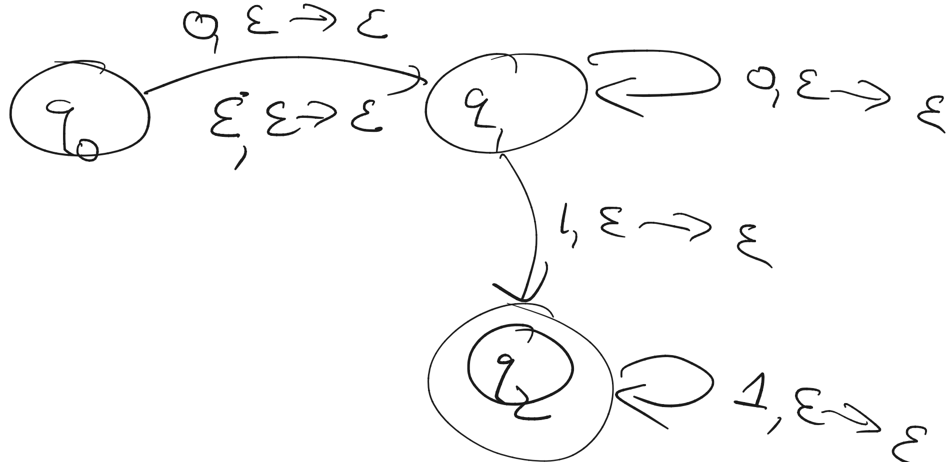
Ex.

NFA:



q_0
q_1
q_2
0
ε
0
1
1

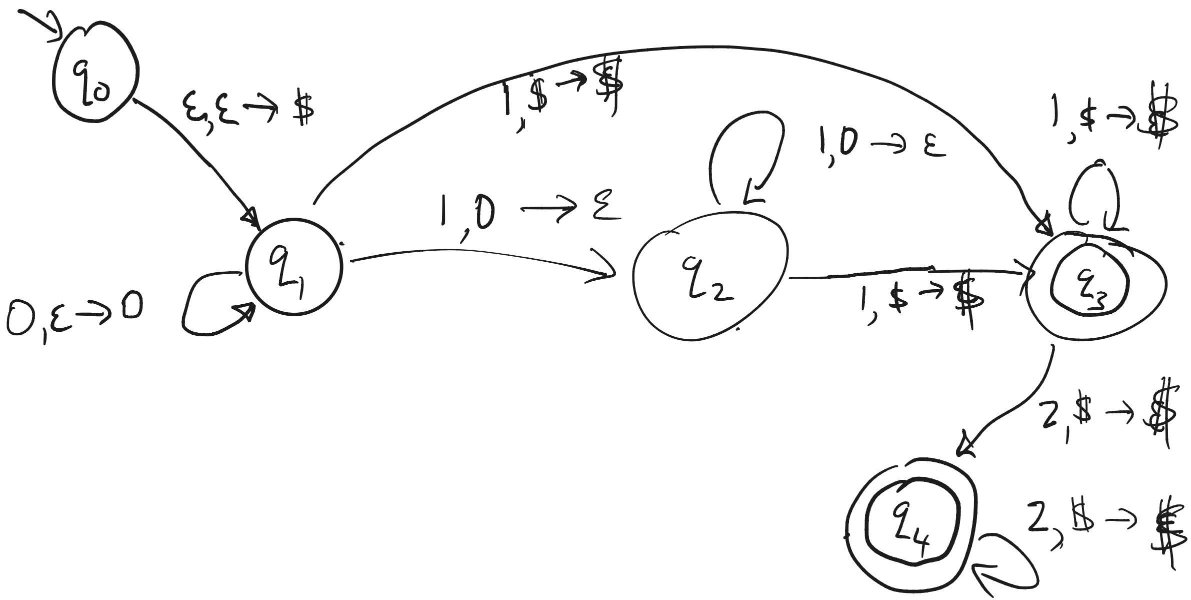
PDA:



q_0
q_1
q_2
$0, \varepsilon \to \varepsilon$
$\varepsilon, \varepsilon \to \varepsilon$
$0, \varepsilon \to \varepsilon$
$1, \varepsilon \to \varepsilon$
$1, \varepsilon \to \varepsilon$

Example 4

$L = \{ 0^i 1^j 2^* \mid i < j \}$ $\Sigma = \{0, 1, 2\}$

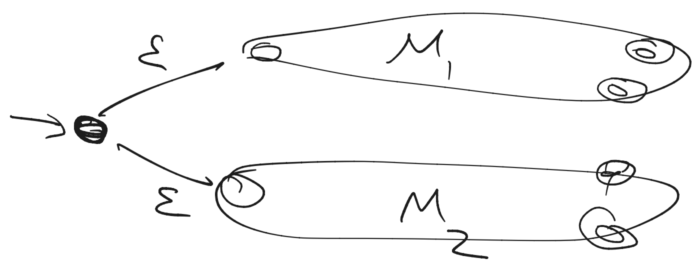


Hints on making PDAs

1. CFL's are closed under union

If M_1 is a PDA for L_1
M_2 " L_2

then can construct M PDA for $L_1 \cup L_2$



2. CFLs NOT closed under complement.

✳ An important difference between CFL's and Regular L's:

	Closed Under Negation	Deterministic = Nondet
Regular	yes	Yes (DFA ≡ NFA)
CFL	No	No (Deterministic PDA ≢ Nondet PDA)

(2) Alternative Characterization of CFL's : Context-Free grammers

<u>Defn</u> A context-free grammar (CFG) is a 4-tuple

$$\mathcal{G} = (V, \Sigma, R, S)$$

V: finite set of variables

Σ: Finite alphabet

R: Finite set of rules of the form $A \to w$, $A \in V$, $w \in (V \cup \Sigma)^*$

$S \in V$ is the start variable

Equivalent Characterization of PDA's : Context-Free grammers

Defn A context-free grammar (CFG) is a 4-tuple

$$\mathcal{G} = (V, \Sigma, R, S)$$

- V: finite set of variables
- Σ: Finite alphabet
- R: Finite set of rules of the form $A \to w$, $A \in V$, $w \in (V \cup \Sigma)^*$
- $S \in V$ is the start variable

Example 1 $\mathcal{G} = (V = \{S\}, \Sigma = \{0,1\}, R, S)$

R :

$S \to 0S1$

$S \to \varepsilon$

(can abbreviate both by $S \to \varepsilon \mid 0S1$)

ex. cant generate 011

ex. generating 0011 : $S \to 0\underline{S}1 \xrightarrow{S \to 0S1} 00\underline{S}11 \to 0011$

Equivalent Characterization of PDA's : Context-Free grammers

Example 1 $\mathcal{G} = (V = \{S\}, \Sigma = \{0,1\}, R, S)$

R : $S \to S$
$S \to 0S1$

(can abbreviate both by $S \to S \mid 0S1$)

Defn For a CFG $\mathcal{G} = (V, \Sigma, R, S)$

Let $u, v, w \in (V \cup \Sigma)^*$, and suppose the rule $A \to w$ is in R. Then we say that $uAv \Rightarrow uwv$ (uAv yields uwv)

If $\exists\, u_1, \dots, u_n \in (V \cup \Sigma^*)$ such that $u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \cdots \Rightarrow u_n \Rightarrow V$

then we say $u \overset{*}{\Rightarrow} v$ (u generates v)

The Language $\mathcal{L}(\mathcal{G})$ generated by $\mathcal{G}$ is

$$L(\mathcal{G}) = \{ v \in \Sigma^* \mid S \overset{*}{\Rightarrow} v \}$$

Equivalent Characterization of PDA's : Context-Free grammers

Example 1 $\mathcal{G} = (V = \{S\}, \Sigma = \{0,1\}, R, S)$

R : $S \to \varepsilon$
$S \to 0S1$

(can abbreviate both by $S \to \varepsilon \mid 0S1$)

$S \to 0S1 \to 00S11 \to 000S111 \to 000111$

So $000111 \in \mathcal{L}(\mathcal{G})$

$\mathcal{L}(\mathcal{G}) = \{0^n 1^n \mid n \geq 0\}$

Example 3 $L = \{ w \mid w = w^R \}$ $\Sigma = \{a, b\}$

$S \to aSa$

$S \to bSa$

$S \to \varepsilon \mid a \mid b$ $S \to aSa \mid bSb \mid \varepsilon \mid a \mid b$

$S \to aSa \to abSba \to abba$

$S \to aSa \to abSba \to ababa$

$S \to bSb$

Example 2

$\mathcal{G} = (V = \{S\}, \Sigma = \{a, b\}, R, S)$

R: $S \to \varepsilon \mid a \mid b$

$S \to aSa \mid bSb$

$$\left.\begin{array}{l} S \to \varepsilon \\ S \to a \\ S \to b \end{array}\right\} \text{ So } \varepsilon, a, b \in \mathcal{L}(\mathcal{G})$$

$S \to aSa \to aaSaa \to aabSbaa \to aabbaa$

$S \to aSa \to aaSaa \to aabSbaa \to aababaa$

Example 2

$L = \{ w \in \{a,b\}^* \mid w = w^R \}$

$$\mathcal{G} = (V = \{S\}, \Sigma = \{a,b\}, R, S)$$

R: $S \to \varepsilon \mid a \mid b$

$S \to aSa \mid bSb$

$$\left.\begin{array}{l} S \to \varepsilon \\ S \to a \\ S \to b \end{array}\right\} \text{ So } \varepsilon, a, b \in \mathcal{L}(\mathcal{G})$$

$S \to aSa \to aaSaa \to aabSbaa \to aabbaa$

$S \to aSa \to aaSaa \to aabSbaa \to aababaa$

Example 3:

$L = \{ w \mid w = w^R \text{ and } |w| \text{ is even} \}$

$G = (V = \{S\}, \Sigma = \{a, b\}, R, S)$

R: $S \to \varepsilon \mid \cancel{a} \mid \cancel{b}$

$S \to aSa \mid bSb$

$S \to \varepsilon$

$\cancel{S \to a}$

$\cancel{S \to b}$

$S \to aSa \to aaSaa \to aabSbaa \to aabbaa$

$\cancel{S \to aSa \to aaSaa \to aabSbaa \to aababaa}$

$L = \{ w \mid w = w^R \text{ and } |w| \text{ is odd} \}$ $S \to aSa \mid bSb \mid a \mid b$

Example 4

$\mathcal{G} = (V = \{E\}, \Sigma = \{a, b, +, *, (,)\}, R, E)$

$R:$ $E \to E + E \mid E * E \mid (E) \mid a \mid b$

Derivation for $a + b * a \in \mathcal{L}(\mathcal{G})$:

$\underline{E} \to E + \underline{E} \to \underline{E} + E * E \to a + \underline{E} * E \to a + b * \underline{E} \to a + b * a$

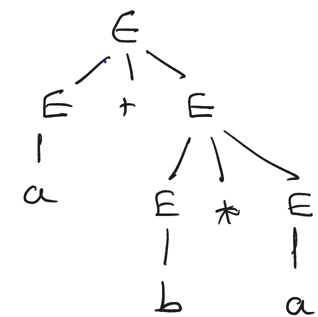
Example 4

$G = (V = \{E\}, \Sigma = \{a, b, +, *, (,)\}, R, E)$

$R:$ $E \to E+E \mid E*E \mid (E) \mid a \mid b$

Derivation #1 for $a+b*a \in \mathcal{L}(G)$:

$E \to E+E \to E+E*E \to a+E*E \to a+b*E \to a+b*a$



Derivation tree

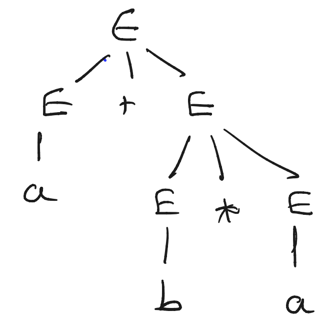
Example 4

$G = (V = \{E\}, \Sigma = \{a, b, +, *, (,)\}, R, E)$

R : $E \to E+E \mid E*E \mid (E) \mid a \mid b$

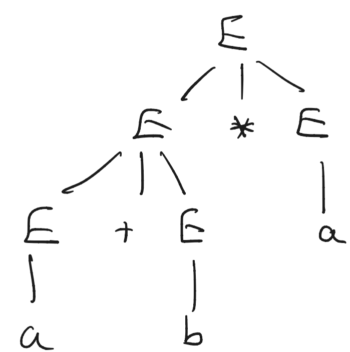
Derivation #1 for a+b*a :

$E \to E+E \to E+E*E \to a+E*E \to a+E*a \to a+b*a$



Derivation #2 for a+b*a :

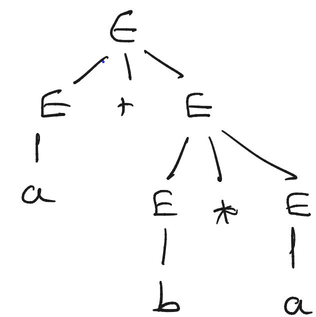
$E \to E*E \to E*a \to E+E*a \to a+E*a \to a+b*a$



Defn. A leftmost derivation is a derivation where at each step, we replace the leftmost variable

⇒ Derivations #1 and #2 were not Leftmost.
The corresponding Leftmost derivations are:

Derivation #1 $\left(E \to E+E \mid E*E \mid (E) \mid a \mid b\right)$



$E \to E + E \to a + E \to a + E * E \to a + b * E \to a + b * a$

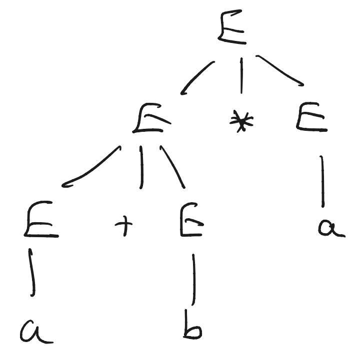
Corresponding Leftmost:

$E \to E + E \to E + E * E \to a + E * E \to a + E * a \to a + b * a$

Defn. A leftmost derivation is a derivation where at each step, we replace the leftmost variable

Derivation #2

$$E \to E * E \to E * a \to E + E * a \to a + E * a \to a + b * a$$



Corresponding Leftmost:

$$E \to E * E \to E + E * E \to a + E * E \to a + b * E \to a + b * a$$

Claim There is a 1-1 correspondence between a derivation tree and a leftmost derivation

Ambiguous vs UnAmbiguous grammars

Defn A CFG $\mathcal{G}$ is ambiguous if there exists some $w \in \mathcal{L}(\mathcal{G})$ such that w has more than one different derivation trees ($\equiv$ more than one leftmost derivation)

Example 4 is ambiguous since we just saw that $w = a+b*a$ has 2 different derivation trees

Ambiguous vs UnAmbiguous grammars

Defn A CFG G is ambiguous if there exists some $w \in \mathcal{L}(G)$ such that w has more than one different derivation trees ($\equiv$ more than one leftmost derivation)

Example 4 G: $E \to E+E \mid E*E \mid (E) \mid a \mid b$ is ambiguous since $w = a+b*a$ has 2 different derivation trees

Define: G' :

$$E \to E+F \mid F$$
$$F \to F*G \mid G$$
$$G \to (E) \mid a \mid b$$

Claim G' is unambiguous, and $\mathcal{L}(G) = \mathcal{L}(G')$

Ambiguous vs UnAmbiguous grammars

Defn A CFG G is *ambiguous* if there exists some $w \in \mathcal{L}(G)$ such that w has more than one different derivation trees ($\equiv$ more than one leftmost derivation)

Defn A context free language L is *inherently ambiguous* if every CFG that generates L is ambiguous.

Ambiguous vs UnAmbiguous grammars

Defn A CFG G is ambiguous if there exists some $w \in \mathcal{L}(G)$ such that w has more than one different derivation trees ($\equiv$ more than one leftmost derivation)

Defn A context free Language L is inherently ambiguous if every CFG that generates L is ambiguous.

Example $L = \{a^n b^n c^m d^m \mid n, m \geq 0\} \cup \{a^n b^m c^n d^m \mid n, m \geq 0\}$ is inherently ambiguous

Ambiguous vs UnAmbiguous grammars

Defn A context free Language L is *inherently ambiguous* if every CFG that generates L is ambiguous.

Example $L = \{a^n b^n c^m d^m \mid n, m \geq 0\} \cup \{a^n b^m c^n d^m \mid n, m \geq 0\}$ is inherently ambiguous

Claim 1 L is a CFL (Prove as an exercise)

Claim 2 (idea): show that any w of the form $a^n b^n c^n d^n$, $n \geq 2$ will always have at least 2 different derivation trees