Theory of Automata and Languages

Pushdown Automata and Context-Free Languages - 1

Fall 2024

Sharif University of Technology

Mehran Moeini Jam



Overview

- Context-free Grammars and Languages
- Derivation (Parse) Trees
- Recognizing the Language of a Grammar
- Ambiguous Grammars and CFLs
- Proof of Unambiguity
- Chomsky Normal Form
- Non-deterministic Pushdown Automata (NPDA)

Context-free Grammars

A context-free grammar G is 4-tuple (V, Σ, R, S) where

- V is a finite set called the variables,
- Σ is a finite set, disjoint from V, called the terminals,
- R is a finite set of rules, with each rule being an ordered set containing a variable as the first element and a string of variables and terminals as the second.
- $S \in V$ is the start variable.

Context-free Grammars

- Consider u, v and w are strings of variables and terminals, and $A \rightarrow w$ is a rule in the grammar, then we say that uAv yields uwv, written $uAv \Rightarrow uwv$.

- We say that u derives v, written $u \overset{*}{\Rightarrow} v$, when either u = v or a sequence $u_1, \ldots, u_k$ exists for $k \geq 0$ and the following holds:

$$u \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k \Rightarrow v$$

Context-free Languages

- The language of a grammar $G = (V, \Sigma, R, S)$ is the set of strings in Σ^* that can be derived from the start variable S, in other words:

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

- The class of all languages that can be generated by a context-free grammar is called the class of context-free languages.

Context-free Grammars

- For every string in a context-free language based on a specific grammar, there exist a derivation (parse) tree that represents how the string can be derived from the grammar.

- The derivation tree is a rooted tree, with the root labeled as the start symbol, internal nodes as variable symbols and leaves as terminals.

Example

- Consider grammar (V, Σ, R, <EXPR>) where

  V = {<EXPR>, <TERM>, <FACTOR>} and

  Σ = { a, +, ×, (,) } and

  the set of rules R is as follows:

  <EXPR> → <EXPR> + <TERM> | <TERM>

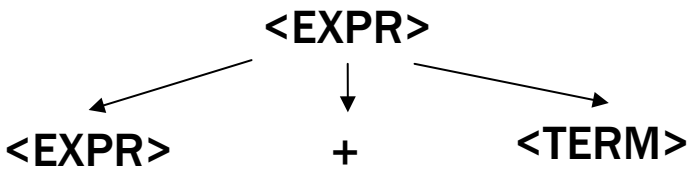
  <TERM> → <TERM> × <FACTOR> | <FACTOR>

  <FACTOR> → (<EXPR>) | a

Example

- The following is the derivation tree for

  the string a + a × a



<EXPR> → <EXPR> + <TERM> | <TERM>

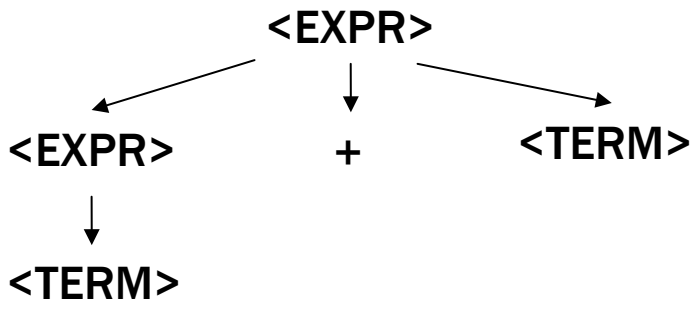
<TERM> → <TERM> × <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR>) | a

Example

- The following is the derivation tree for

  the string a + a × a



<EXPR> → <EXPR> + <TERM> | <TERM>

<TERM> → <TERM> × <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR>) | a

Example

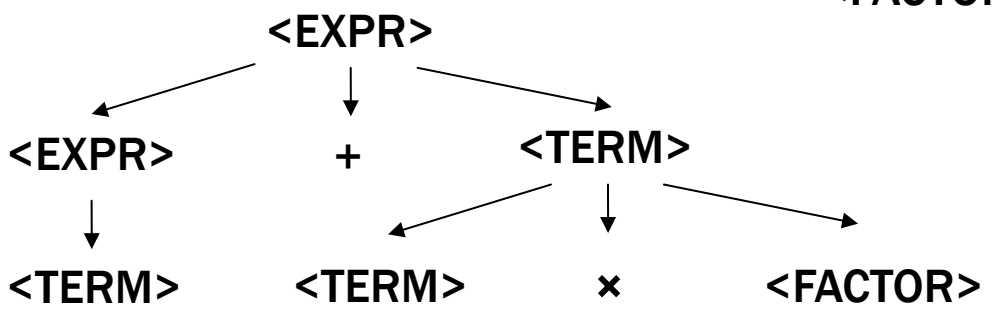
- The following is the derivation tree for

  the string a + a × a

<EXPR> → <EXPR> + <TERM> | <TERM>

<TERM> → <TERM> × <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR>) | a



Example

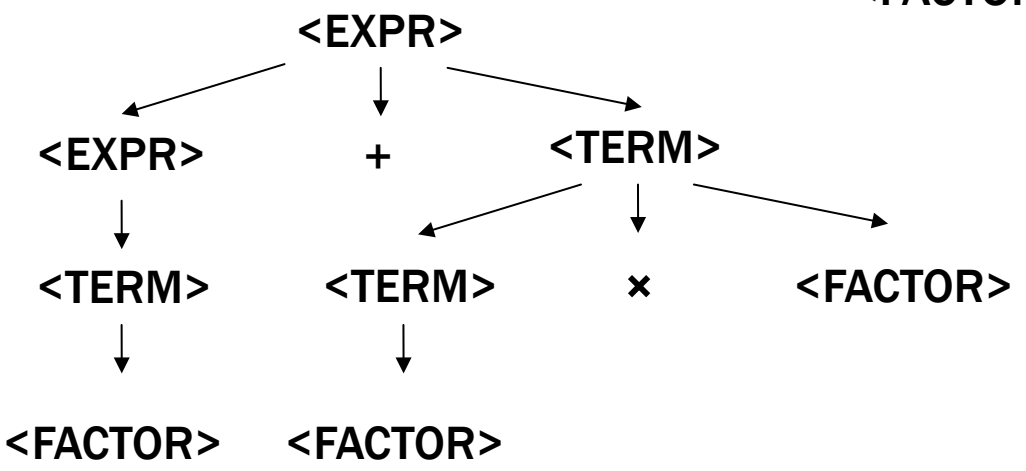
- The following is the derivation tree for

  the string a + a × a

<EXPR> → <EXPR> + <TERM> | <TERM>

<TERM> → <TERM> × <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR>) | a



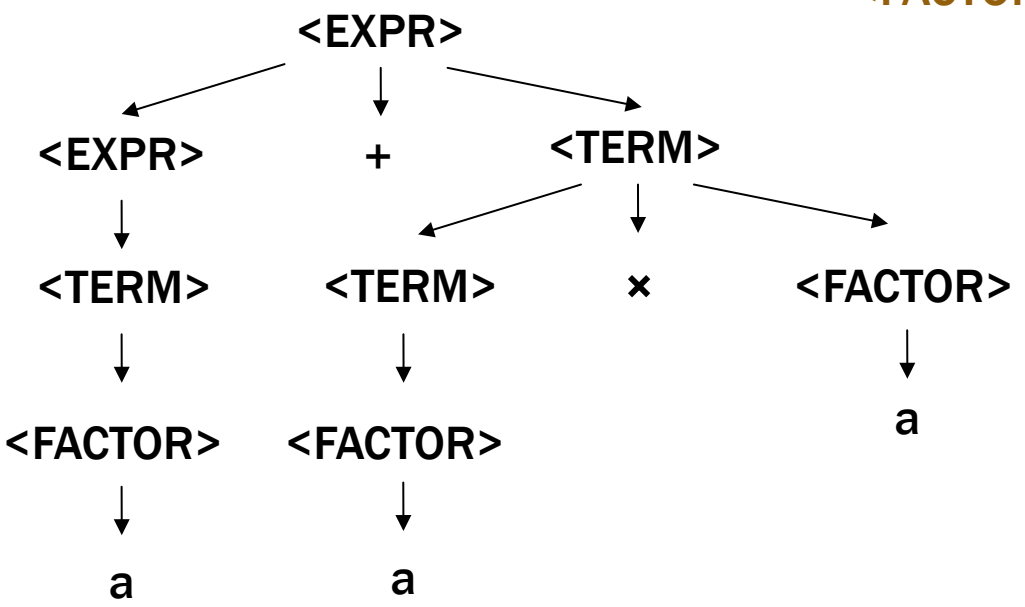
Example

- The following is the derivation tree for the string a + a × a

<EXPR> → <EXPR> + <TERM> | <TERM>

<TERM> → <TERM> x <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR>) | a



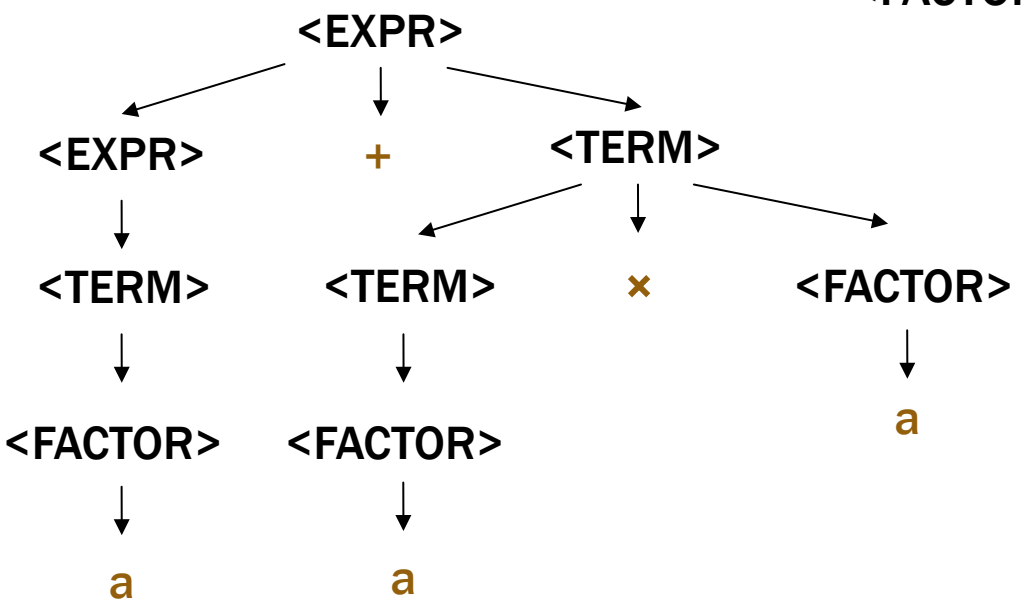
Example

- The following is the derivation tree for the string a + a × a

<EXPR> → <EXPR> + <TERM> | <TERM>

<TERM> → <TERM> x <FACTOR> | <FACTOR>

<FACTOR> → (<EXPR>) | a



Recognizing the Language of a Grammar

- Consider the following grammar G:

$S \rightarrow aB \mid bA$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

- The language L(G) of the grammar is the set of all words in Σ^+ consisting of an equal number of a's and b's.

Recognizing the Language of a Grammar

- The language L(G) of the grammar is

  the set of all words in Σ^+ consisting of an

  equal number of a's and b's.

$S \rightarrow aB \mid bA$

$A \rightarrow bAA \mid aS \mid a$

$B \rightarrow aBB \mid bS \mid b$

- To prove this statement, we need to prove the following three statements:

1. $\forall w \in \Sigma^* \; (\; S \overset{*}{\Rightarrow} w \quad \leftrightarrow \quad$ w consists of an equal number of a's and b's $)$
2. $\forall w \in \Sigma^* \; (\; A \overset{*}{\Rightarrow} w \quad \leftrightarrow \quad$ w has one more a than it has b's $)$
3. $\forall w \in \Sigma^* \; (\; B \overset{*}{\Rightarrow} w \quad \leftrightarrow \quad$ w has one more b than it has a's $)$

Recognizing the Language of a Grammar

1. $\forall w \in \Sigma^* (S \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad w \text{ consists of an equal number of a's and b's })$ $\quad S \rightarrow aB \mid bA$
2. $\forall w \in \Sigma^* (A \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad w \text{ has one more a than it has b's })$ $\quad A \rightarrow bAA \mid aS \mid a$
3. $\forall w \in \Sigma^* (B \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad w \text{ has one more b than it has a's })$ $\quad B \rightarrow aBB \mid bS \mid b$

Consider $|w| = 1$, then (1) and (2) and (3) are all true, by knowing that:

- No string of length one is derivable from S.
- Every rule except $A \rightarrow a$ and $B \rightarrow b$ increases the length of the string, and no strings of length one, other than **a** and **b**, are derivable from those two rules.

Recognizing the Language of a Grammar

1. $\forall w \in \Sigma^*$ ($S \overset{*}{\Rightarrow} w$ $\rightleftarrows$ w consists of an equal number of a's and b's) $\quad S \rightarrow aB \mid bA$
2. $\forall w \in \Sigma^*$ ($A \overset{*}{\Rightarrow} w$ $\rightleftarrows$ w has one more a than it has b's) $\quad A \rightarrow bAA \mid aS \mid a$
3. $\forall w \in \Sigma^*$ ($B \overset{*}{\Rightarrow} w$ $\rightleftarrows$ w has one more b than it has a's) $\quad B \rightarrow aBB \mid bS \mid b$

Consider the above statements are true for $|w| = k$, then for $|w| = k + 1$:

- If $S \overset{*}{\Rightarrow} w$ then the derivation must begin with either $S \rightarrow aB$ or $S \rightarrow bA$. In the first case, w is of the form aw_1 where $|w_1| = k$, so by the **third** statement, w_1 has one more **b** than it has **a**'s, and adding up the **a** at the beginning, we conclude that w has equal number of **a**'s and **b**'s. A Similar argument is true for the second rule.

Recognizing the Language of a Grammar

1. $\forall w \in \Sigma^* (S \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad$ w consists of an equal number of a's and b's $)$ — $S \rightarrow aB \mid bA$
2. $\forall w \in \Sigma^* (A \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad$ w has one more a than it has b's $)$ — $A \rightarrow bAA \mid aS \mid a$
3. $\forall w \in \Sigma^* (B \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad$ w has one more b than it has a's $)$ — $B \rightarrow aBB \mid bS \mid b$

Consider the above statements are true when $|w| = k$, then for $|w| = k + 1$:

- If w consist of an equal number of **a**'s and **b**'s, then either its first symbol is **a** or it is **b**. Assume it's **a**, then it is of the form $\mathbf{a}w_1$ where $|w_1| = k$, and since w_1 has one more b than a, it can be derived from B, and since $S \Rightarrow aB \overset{*}{\Rightarrow} aw_1 = w$, then we can conclude that $S \overset{*}{\Rightarrow} w$. A similar argument can be provided when w starts with **b**.

Recognizing the Language of a Grammar

1. $\forall w \in \Sigma^* (S \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad w \text{ consists of an equal number of a's and b's })$ $\quad S \rightarrow aB \mid bA$
2. $\forall w \in \Sigma^* (A \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad w \text{ has one more a than it has b's })$ $\quad A \rightarrow bAA \mid aS \mid a$
3. $\forall w \in \Sigma^* (B \overset{*}{\Rightarrow} w \quad \rightleftarrows \quad w \text{ has one more b than it has a's })$ $\quad B \rightarrow aBB \mid bS \mid b$

- Our proof is not yet done.

1. Statement (2) should be proved in a similar manner for $|w| = k + 1$
2. Statement (3) should be proved in a similar manner for $|w| = k + 1$

Ambiguous Grammars

- A grammar is called ambiguous where there exists a string ω in the language of the grammar that can be represented by two distinct derivation trees.

- The above sentence can be stated in another ways as well.

Rightmost and Leftmost Derivations

- In rightmost derivation, the derivation is based on expanding the rightmost variable each time to achieve the final string.

<EXPR> ⇒ <TERM> ⇒ <TERM> × <FACTOR> ⇒ <TERM> × a ⇒ <FACTOR> × a ⇒ a × a

- In leftmost derivation, the derivation is based on expanding the leftmost variable each time to achieve the final string.

<EXPR> ⇒ <TERM> ⇒ <TERM> × <FACTOR> ⇒ <FACTOR> × <FACTOR> ⇒ a × <FACTOR> ⇒ a × a

Ambiguous Grammars

- A grammar is called ambiguous where there exists a string ω in the language of the grammar that can be represented by two distinct derivation trees.

- The above sentence can be stated in another ways as well.

- A grammar is also called ambiguous where there exist a string ω that can be generated by two distinct rightmost or two distinct leftmost derivations.

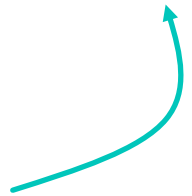
Pitfall

- A grammar is called ambiguous where there exists a string ω in the language of the grammar that can be represented by two distinct derivation trees.

- The above sentence can be stated in another ways as well.

- A grammar is also called ambiguous where there exist a string ω that can be generated by two distinct rightmost or two distinct leftmost derivations.

Not a rightmost and a leftmost derivation



Example

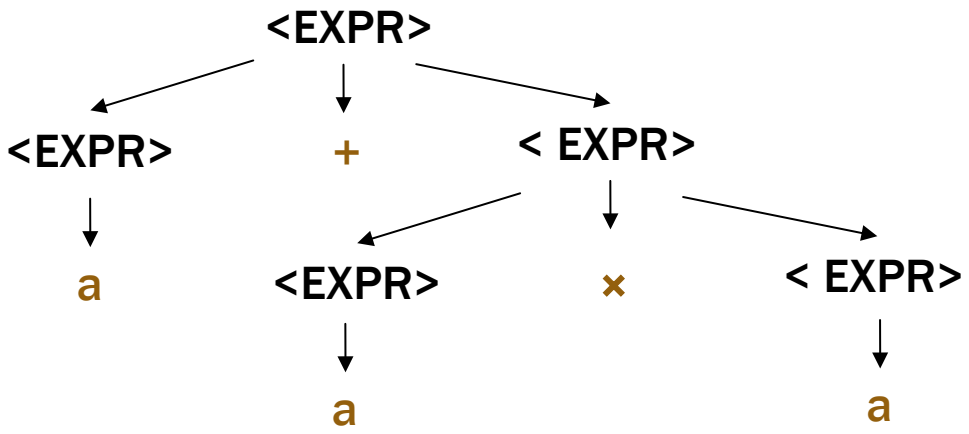
- Consider the grammar

$$\langle EXPR \rangle \rightarrow \langle EXPR \rangle + \langle EXPR \rangle \mid \langle EXPR \rangle \times \langle EXPR \rangle \mid (\langle EXPR \rangle) \mid a$$

- The string a + a × a can be generated by two different derivation trees as shown in the next slide.

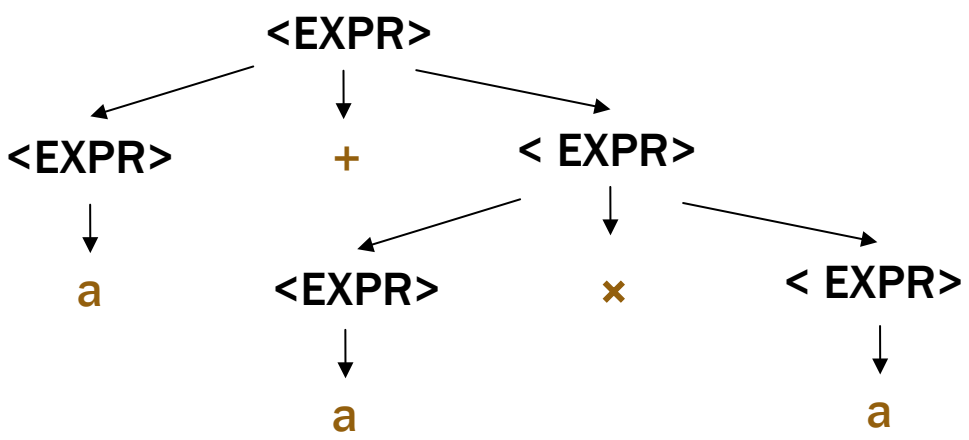
Example

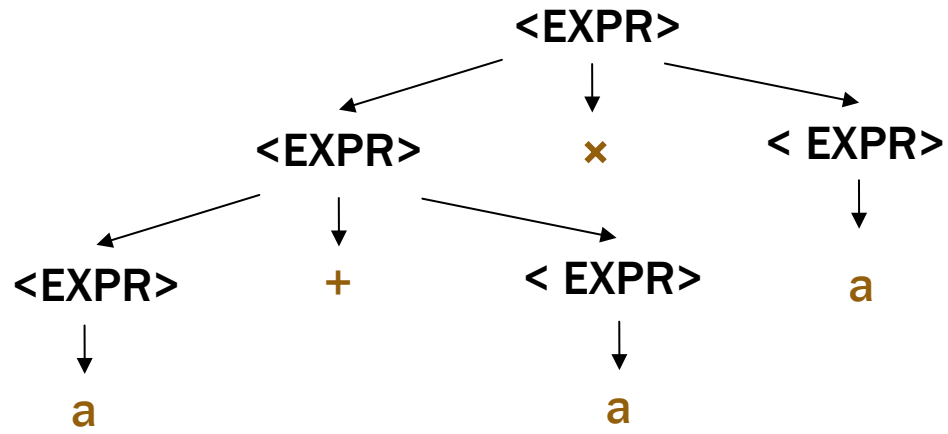
<EXPR> → <EXPR> + <EXPR> | <EXPR> × <EXPR> | (<EXPR>) | a



Example

<EXPR> → <EXPR> + <EXPR> | <EXPR> × <EXPR> | (<EXPR>) | a





Ambiguous Languages

- A context-free language is said to be inherently ambiguous when no unambiguous context-free grammar exists that generates the same set of strings.

- Consider for example the following language, it can be proved (Hoprocft, Ullman) that no unambiguous context-free grammar exists for this language:

$$L = \{ A^n b^n c^m d^m \mid n \geq 1, m \geq 1 \} \cup \{a^n b^m c^m d^n \mid n \geq 1, m \geq 1\}$$

- In L, infinitely many strings of the form $a^n b^n c^n d^n$ have two distinct parse trees.

Ambiguous Languages

- It can be proved that there is no algorithm that given an arbitrary context-free grammar, it can determine whether the grammar is ambiguous or not.

- The same can be said regarding the context-free languages, where there is no algorithm to determine whether it has a non-ambiguous grammar or not (i.e. deciding whether it is inherently ambiguous).

Proof of Unambiguity

- Consider the following grammar G :

  $E \rightarrow E + T \mid T$

  $T \rightarrow T \times F \mid F$

  $F \rightarrow a \mid (E)$

- To prove that the above grammar is unambiguous, we show that for every $w \in \Sigma^*$ that is derivable from each variable, there only exists one leftmost (or one rightmost) derivation.

Proof of Unambiguity

- The basis part, for w = a, is easy,

  because for each of the three variables,

  a has exactly one derivation from that variable.

$E \rightarrow E + T \mid T$

$T \rightarrow T \times F \mid F$

$F \rightarrow a \mid (E)$

- Try and show the above statement by providing the unique parse trees, where

  each start from one of the three variables.

Proof of Unambiguity

- For the inductive part, we divide the strings

  in the following (arbitrary) way:

$E \rightarrow E + T \mid T$

$T \rightarrow T \times F \mid F$

$F \rightarrow a \mid (E)$

1. $w \in \Sigma^*$ has at least one + outside parentheses.
2. $w \in \Sigma^*$ has at least one x outside parentheses, and no + as such.
3. $w \in \Sigma^*$ has no + or x outside parentheses.

Proof of Unambiguity

- Consider the statements to be true for $|w| \leq k$, we have to show that they are also true for $|w| = k + 1$

$E \rightarrow E + T \mid T$

$T \rightarrow T \times F \mid F$

$F \rightarrow a \mid (E)$

1. $w \in \Sigma^*$ has at least one + outside parentheses.

T and F can't derive such a string, so every derivation in this form must only begin with the rule $E \rightarrow E + T$, so we conclude that $E \Rightarrow E + T \overset{*}{\Rightarrow} y + T \overset{*}{\Rightarrow} y + z = w$ where the length of y and z is less than or equal to k, and they have only one derivation. Done!

Proof of Unambiguity

- Consider the statements to be true for $|w| \leq k$, we have to show that they are also true for $|w| = k + 1$

$E \rightarrow E + T \mid T$

$T \rightarrow T \times F \mid F$

$F \rightarrow a \mid (E)$

2. $w \in \Sigma^*$ has at least one x outside parentheses, and no + as such.

F can't derive such a string, so every derivation in this form must only begin with

$E \rightarrow T$ (for E) or $T \rightarrow T \times F$ (for T), so we conclude that $E \Rightarrow T \Rightarrow T \times F \overset{*}{\Rightarrow} y \times F \overset{*}{\Rightarrow} y \times z = w$

where the length of y and z is less than or equal to k, and they have only one derivation.

Proof of Unambiguity

- Consider the statements to be true for $|w| \leq k$, we have to show that they are also true for $|w| = k + 1$

$E \rightarrow E + T \mid T$

$T \rightarrow T \times F \mid F$

$F \rightarrow a \mid (E)$

3. $w \in \Sigma^*$ has no + or x outside parentheses.

Such a string can be derived from each one of the variables, where for F it must begin with $F \rightarrow (E)$, for T it must begin with $T \rightarrow F$ and for E it must begin with $E \rightarrow T$.

Because in each case, we have $(E) \overset{*}{\Rightarrow} (y)$ where $|y| \leq k$, the proof concludes.

Chomsky Normal Form

- A context-free grammar is in Chomsky Normal Form if every rule is of the form

$$A \rightarrow BC$$

$$A \rightarrow a$$

where **a** is any terminal and A, B, and C are any variables.

- By the above definition, no language that contains ε can be represented by a grammar in Chomsky normal form. So, the definition needs to be changed.

Chomsky Normal Form: Re-definition

- A context-free grammar is in Chomsky Normal Form if every rule is of the form

$$A \rightarrow BC$$

$$A \rightarrow a \qquad\qquad S_0 \rightarrow \varepsilon$$

where **a** is any terminal, S_0 is the start variable and A, B, and C are variables, with the condition that B and C can not be the start variable.

Notes

- In the last definition, we preserve the ability to generate the empty string, but also prohibit using the start variable at right-hand side (RHS) of the rules.

- The main reason is that we want to avoid rules such as $A \rightarrow S_0S_0$ which, combined with the rule $S_0 \rightarrow \varepsilon$ allows the empty string to be derived from variables other than the start variable. This is unsatisfactory due to other reasons that we will see later.

Convert CFGs to Chomsky Normal Form

- Any context-free language can be generated by a context-free grammar in Chomsky normal form.

- To prove the above statement, we can show that any grammar G can be converted into the normal form. The conversion has several stages wherein rules that violate the conditions are replaced with equivalent ones that are satisfactory.

Chomsky Normal Form

- Any context-free language can be generated by a context-free grammar in Chomsky normal form.

1. Insert a new start variable S_0 and the rule $S_0 \rightarrow S$
2. Take care of all the ε-rules, i.e., rules in the form $A \rightarrow \varepsilon$ where $A \neq S_0$
3. Take care of all the unit rules, i.e., rules in the form $A \rightarrow B$
4. Convert all the remaining rules into the proper form

Removing ε-rules

- The second step is taking care of all the ε-rules, i.e. the rules in the form

$$A \rightarrow \varepsilon$$

where A is a variable other than the start symbol.

- We first remove the above ε-rule, and then, for each occurrence of an A on the right-hand side of a rule, we add new rules with that occurrence deleted. In other words, if $R \rightarrow uAv$ is a rule in which $u, v \in (V \cup \Sigma)^*$, we add rule $R \rightarrow uv$.

Removing ε-rules

- We must note that if we have both the rules $R \rightarrow A$ and $A \rightarrow \varepsilon$, a new ε-rule

  $R \rightarrow \varepsilon$ must be added to the rule set, to be handled later like the others.

- If the rule $R \rightarrow \varepsilon$ had previously been in the rule set and was handled accordingly,

  no such new rule is needed again.

Pitfall

- In removing ε-rules, when the variable A has n occurences in the right hand side of a rule, $2^n - 1$ new rules must be created, each time removing one set of occurences.

- For example, if we have the rule $R \rightarrow uAvAw$ where there are two occurences of A, we must add the rules $R \rightarrow uvAw$, $R \rightarrow uAvw$, and $R \rightarrow uvw$.

Removing unit rules

- The third step is taking care of all the unit rules, i.e. the rules in the form

$$A \to B$$

where both A and B are variables.

- We first remove the above unit rule, and then, whenever a rule $B \to u$ appears, where $u \in (V \cup \Sigma)^*$, we add a new rule $A \to u$.

Final conversion to achieve CNF

- The final step is to convert all the remaining rules into the proper form.

- In order to do so, we replace each rule $A \rightarrow u_1u_2...u_k$ where $k \geq 3$ and $u_i \in (V \cup \Sigma)$ for every $1 \leq i \leq k$, with the rules $A \rightarrow u_1A_1$, $A_1 \rightarrow u_2A_2$ and so forth.

- We finally replace each terminal u_i in the above rules with a new variable U_i and a new rule $U_i \rightarrow u_i$ and we are done.

Example

- Consider the following grammar:

$$S \rightarrow ASA \mid aB$$

$$A \rightarrow B \mid S$$

$$B \rightarrow b \mid \varepsilon$$

Example

- Make the new start symbol:

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \varepsilon$

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \varepsilon$

Example

- Remove the epsilon rule $B \rightarrow \varepsilon$:

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB$

$A \rightarrow B \mid S$

$B \rightarrow b \mid \varepsilon$

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a$

$A \rightarrow B \mid S \mid \varepsilon$

$B \rightarrow b$

Example

- Remove the epsilon rule $A \rightarrow \varepsilon$:

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a$

$A \rightarrow B \mid S \mid \varepsilon$

$B \rightarrow b$



$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$

$A \rightarrow B \mid S$

$B \rightarrow b$

Example

- Remove the unit rule $S \rightarrow S$:

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS \mid S$

$A \rightarrow B \mid S$

$B \rightarrow b$



$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow B \mid S$

$B \rightarrow b$

Example

- Remove the unit rule $S_0 \rightarrow S$:

$S_0 \rightarrow S$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow B \mid S$

$B \rightarrow b$



$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow B \mid S$

$B \rightarrow b$

Example

- Remove the unit rule $A \rightarrow B$:

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow B \mid S$

$B \rightarrow b$

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow S \mid b$

$B \rightarrow b$

Example

- Remove the unit rule $A \rightarrow S$:

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow S \mid b$

$B \rightarrow b$



$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$

Example

- Convert the remaining rules:

$S_0 \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$S \rightarrow ASA \mid aB \mid a \mid SA \mid AS$

$A \rightarrow b \mid ASA \mid aB \mid a \mid SA \mid AS$

$B \rightarrow b$



$S_0 \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$

$S \rightarrow AA_1 \mid UB \mid a \mid SA \mid AS$

$A \rightarrow b \mid AA_1 \mid UB \mid a \mid SA \mid AS$

$A_1 \rightarrow SA$

$U \rightarrow a$

$B \rightarrow b$

Done!

Non-deterministic Pushdown Automata (NPDA)

Next Set of Slides!

