BBM401-Lecture 9: Context-Free Grammars

Lecturer: Lale Özkahya

Resources for the presentation:
https://courses.engr.illinois.edu/cs373/fa2010/

Context-Free Grammars

Definition

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

Context-Free Grammars

Definition

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

- V is a finite set of variables/non-terminals.

Context-Free Grammars

Definition

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

- V is a finite set of variables/non-terminals.
- Σ is a finite set of terminals. Σ is disjoint from V.

Context-Free Grammars

Definition

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

- V is a finite set of variables/non-terminals.
- Σ is a finite set of terminals. Σ is disjoint from V.
- R is a finite set of rules or productions of the form $A \rightarrow \alpha$ where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$

Context-Free Grammars

Definition

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

- V is a finite set of variables/non-terminals.
- Σ is a finite set of terminals. Σ is disjoint from V.
- R is a finite set of rules or productions of the form $A \rightarrow \alpha$ where $A \in V$ and $\alpha \in (V \cup \Sigma)^*$
- $S \in V$ is the start symbol

Example: Palindromes

Example

A string w is a palindrome if $w = w^R$.

Example: Palindromes

Example

A string w is a palindrome if $w = w^R$.
$G_{\mathrm{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where R is

$$
\begin{aligned}
S &\rightarrow \epsilon \\
S &\rightarrow 0 \\
S &\rightarrow 1 \\
S &\rightarrow 0S0 \\
S &\rightarrow 1S1
\end{aligned}
$$

Example: Palindromes

Example

A string w is a palindrome if $w = w^R$.

$G_{\mathrm{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where R is

$$\begin{aligned} S &\rightarrow \epsilon \\ S &\rightarrow 0 \\ S &\rightarrow 1 \\ S &\rightarrow 0S0 \\ S &\rightarrow 1S1 \end{aligned}$$

Or more briefly, $R = \{S \rightarrow \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}$

Language of a CFG

Derivations

Expand the start symbol using one of its rules. Further expand the resulting string by expanding one of the variables in the string, by the RHS of one of its rules. Repeat until you get a string of terminals.

Language of a CFG

Derivations

Expand the start symbol using one of its rules. Further expand the resulting string by expanding one of the variables in the string, by the RHS of one of its rules. Repeat until you get a string of terminals. For the grammar $G_{\text{pal}} = (\{S\}, \{0,1\}, \{S \rightarrow \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}, S)$ we have

$$S \Rightarrow 0S0 \Rightarrow 00S00 \Rightarrow 001S100 \Rightarrow 0010100$$

Formal Definition

Definition

Let $G = (V, \Sigma, R, S)$ be a CFG. We say $\alpha A \beta \Rightarrow_G \alpha\gamma\beta$, where $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$ and $A \in V$ if $A \rightarrow \gamma$ is a rule of G.

Formal Definition

Definition

Let $G = (V, \Sigma, R, S)$ be a CFG. We say $\alpha A \beta \Rightarrow_G \alpha\gamma\beta$, where $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$ and $A \in V$ if $A \rightarrow \gamma$ is a rule of G.
We say $\alpha \overset{*}{\Rightarrow}_G \beta$ if either $\alpha = \beta$ or there are $\alpha_0, \alpha_1, \ldots \alpha_n$ such that

$$\alpha = \alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \cdots \Rightarrow_G \alpha_n = \beta$$

Formal Definition

Definition

Let $G = (V, \Sigma, R, S)$ be a CFG. We say $\alpha A\beta \Rightarrow_G \alpha\gamma\beta$, where $\alpha, \beta, \gamma \in (V \cup \Sigma)^*$ and $A \in V$ if $A \to \gamma$ is a rule of G.
We say $\alpha \overset{*}{\Rightarrow}_G \beta$ if either $\alpha = \beta$ or there are $\alpha_0, \alpha_1, \ldots \alpha_n$ such that

$$\alpha = \alpha_0 \Rightarrow_G \alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \cdots \Rightarrow_G \alpha_n = \beta$$

Notation

When G is clear from the context, we will write $\Rightarrow$ and $\overset{*}{\Rightarrow}$ instead of $\Rightarrow_G$ and $\overset{*}{\Rightarrow}_G$.

Context-Free Language

Definition

The language of CFG $G = (V, \Sigma, R, S)$, denoted $L(G)$ is the collection of strings over the terminals derivable from S using the rules in R. In other words,

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

Context-Free Language

Definition

The language of CFG $G = (V, \Sigma, R, S)$, denoted $L(G)$ is the collection of strings over the terminals derivable from S using the rules in R. In other words,

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

Definition

A language L is said to be context-free if there is a CFG G such that $L = L(G)$.

Palindromes Revisited

Recall, $L_{\mathrm{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}$ is the language of palindromes.

Palindromes Revisited

Recall, $L_{\mathrm{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}$ is the language of palindromes.
Consider $G_{\mathrm{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where $R = \{S \rightarrow \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}$

Palindromes Revisited

Recall, $L_{\text{pal}} = \{w \in \{0,1\}^* \mid w = w^R\}$ is the language of palindromes.
Consider $G_{\text{pal}} = (\{S\}, \{0,1\}, R, S)$ defines palindromes over $\{0,1\}$, where $R = \{S \rightarrow \epsilon \mid 0 \mid 1 \mid 0S0 \mid 1S1\}$

Proposition

$L(G_{\text{pal}}) = L_{\text{pal}}$

Proving Correctness of CFG

$L_{\mathrm{pal}} \subseteq L(G_{\mathrm{pal}})$

Proof.

Let $w \in L_{\mathrm{pal}}$. We prove that $S \overset{*}{\Rightarrow} w$

Proving Correctness of CFG

$L_{\mathrm{pal}} \subseteq L(G_{\mathrm{pal}})$

Proof.

Let $w \in L_{\mathrm{pal}}$. We prove that $S \stackrel{*}{\Rightarrow} w$ by induction on $|w|$.

Proving Correctness of CFG

$L_{\mathrm{pal}} \subseteq L(G_{\mathrm{pal}})$

Proof.

Let $w \in L_{\mathrm{pal}}$. We prove that $S \stackrel{*}{\Rightarrow} w$ by induction on $|w|$.

- Base Cases: If $|w| = 0$ or $|w| = 1$ then $w = \epsilon$ or 0 or 1. And $S \to \epsilon \mid 0 \mid 1$.

Proving Correctness of CFG

$L_{\mathrm{pal}} \subseteq L(G_{\mathrm{pal}})$

Proof.

Let $w \in L_{\mathrm{pal}}$. We prove that $S \overset{*}{\Rightarrow} w$ by induction on $|w|$.

- Base Cases: If $|w| = 0$ or $|w| = 1$ then $w = \epsilon$ or 0 or 1. And $S \to \epsilon \mid 0 \mid 1$.
- Induction Step: If $|w| \geq 2$ and $w = w^R$ then it must begin and with the same symbol.

Proving Correctness of CFG

$L_{\mathrm{pal}} \subseteq L(G_{\mathrm{pal}})$

Proof.

Let $w \in L_{\mathrm{pal}}$. We prove that $S \overset{*}{\Rightarrow} w$ by induction on $|w|$.

- Base Cases: If $|w| = 0$ or $|w| = 1$ then $w = \epsilon$ or 0 or 1. And $S \rightarrow \epsilon \mid 0 \mid 1$.
- Induction Step: If $|w| \geq 2$ and $w = w^R$ then it must begin and with the same symbol. Let $w = 0x0$.

Proving Correctness of CFG

$L_{\mathrm{pal}} \subseteq L(G_{\mathrm{pal}})$

Proof.

Let $w \in L_{\mathrm{pal}}$. We prove that $S \overset{*}{\Rightarrow} w$ by induction on $|w|$.

- Base Cases: If $|w| = 0$ or $|w| = 1$ then $w = \epsilon$ or 0 or 1. And $S \rightarrow \epsilon \mid 0 \mid 1$.
- Induction Step: If $|w| \geq 2$ and $w = w^R$ then it must begin and with the same symbol. Let $w = 0x0$. Now, $w^R = 0x^R0 = w = 0x0$; thus, $x^R = x$.

Proving Correctness of CFG

$L_{\mathrm{pal}} \subseteq L(G_{\mathrm{pal}})$

Proof.

Let $w \in L_{\mathrm{pal}}$. We prove that $S \overset{*}{\Rightarrow} w$ by induction on $|w|$.

- Base Cases: If $|w| = 0$ or $|w| = 1$ then $w = \epsilon$ or 0 or 1. And $S \rightarrow \epsilon \mid 0 \mid 1$.
- Induction Step: If $|w| \geq 2$ and $w = w^R$ then it must begin and with the same symbol. Let $w = 0x0$. Now, $w^R = 0x^R0 = w = 0x0$; thus, $x^R = x$. By induction hypothesis, $S \overset{*}{\Rightarrow} x$. Hence $S \Rightarrow 0S0 \overset{*}{\Rightarrow} 0x0$. If $w = 1x1$ the argument is similar.

$\cdots\longrightarrow$

Proving Correctness of CFG

$L_{\mathrm{pal}} \supseteq L(G_{\mathrm{pal}})$

Proof (contd).

Let $w \in L(G)$, i.e., $S \stackrel{*}{\Rightarrow} w$. We will show $w \in L_{\mathrm{pal}}$

Proving Correctness of CFG

$L_{\rm pal} \supseteq L(G_{\rm pal})$

Proof (contd).

Let $w \in L(G)$, i.e., $S \stackrel{*}{\Rightarrow} w$. We will show $w \in L_{\rm pal}$ by induction on the number of derivation steps.

Proving Correctness of CFG

$L_{\mathrm{pal}} \supseteq L(G_{\mathrm{pal}})$

Proof (contd).

Let $w \in L(G)$, i.e., $S \overset{*}{\Rightarrow} w$. We will show $w \in L_{\mathrm{pal}}$ by induction on the number of derivation steps.

- Base Case: If the derivation has only one step then the derivation must be $S \Rightarrow \epsilon$, $S \Rightarrow 0$ or $S \Rightarrow 1$. Thus $w = \epsilon$ or 0 or 1 and is in L_{Pal}.

Proving Correctness of CFG

$L_{\mathrm{pal}} \supseteq L(G_{\mathrm{pal}})$

Proof (contd).

Let $w \in L(G)$, i.e., $S \stackrel{*}{\Rightarrow} w$. We will show $w \in L_{\mathrm{pal}}$ by induction on the number of derivation steps.

- Base Case: If the derivation has only one step then the derivation must be $S \Rightarrow \epsilon$, $S \Rightarrow 0$ or $S \Rightarrow 1$. Thus $w = \epsilon$ or 0 or 1 and is in L_{Pal}.
- Induction Step: Consider an $(n+1)$-step derivation of w. It must be of the form $S \Rightarrow 0S0 \stackrel{*}{\Rightarrow} 0x0 = w$ or $S \Rightarrow 1S1 \stackrel{*}{\Rightarrow} 1x1 = w$.

Proving Correctness of CFG

$L_{\mathrm{pal}} \supseteq L(G_{\mathrm{pal}})$

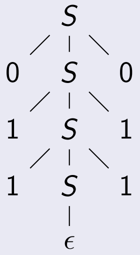
Proof (contd).

Let $w \in L(G)$, i.e., $S \stackrel{*}{\Rightarrow} w$. We will show $w \in L_{\mathrm{pal}}$ by induction on the number of derivation steps.

- **Base Case:** If the derivation has only one step then the derivation must be $S \Rightarrow \epsilon$, $S \Rightarrow 0$ or $S \Rightarrow 1$. Thus $w = \epsilon$ or 0 or 1 and is in L_{Pal}.
- **Induction Step:** Consider an $(n+1)$-step derivation of w. It must be of the form $S \Rightarrow 0S0 \stackrel{*}{\Rightarrow} 0x0 = w$ or $S \Rightarrow 1S1 \stackrel{*}{\Rightarrow} 1x1 = w$. In either case $S \stackrel{*}{\Rightarrow} x$ in n-steps. Hence $x \in L_{\mathrm{Pal}}$ and so $w = w^R$. □

Parse Trees

For CFG $G = (V, \Sigma, R, S)$, a parse tree (or derivation tree) of G is a tree satisfying the following conditions:

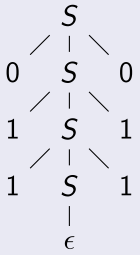


Example Parse Tree with yield 011110

Parse Trees

For CFG $G = (V, \Sigma, R, S)$, a parse tree (or derivation tree) of G is a tree satisfying the following conditions:

- Each interior node is labeled by a variable in V

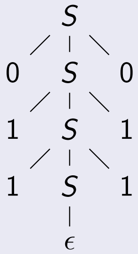


Example Parse Tree with yield 011110

Parse Trees

For CFG $G = (V, \Sigma, R, S)$, a parse tree (or derivation tree) of G is a tree satisfying the following conditions:

- Each interior node is labeled by a variable in V
- Each leaf is labeled by either a variable, a terminal or ϵ; a leaf labeled by ϵ must be the only child of its parent.

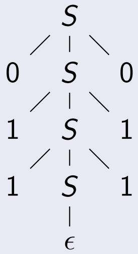


Example Parse Tree with yield 011110

Parse Trees

For CFG $G = (V, \Sigma, R, S)$, a parse tree (or derivation tree) of G is a tree satisfying the following conditions:

- Each interior node is labeled by a variable in V
- Each leaf is labeled by either a variable, a terminal or ϵ; a leaf labeled by ϵ must be the only child of its parent.
- If an interior node labeled by A with children labeled by $X_1, X_2, \ldots X_k$ (from the left), then $A \rightarrow X_1 X_2 \cdots X_k$ must be a rule.

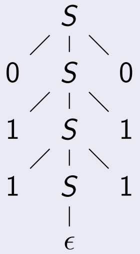


Example Parse Tree with yield 011110

Parse Trees

For CFG $G = (V, \Sigma, R, S)$, a parse tree (or derivation tree) of G is a tree satisfying the following conditions:

- Each interior node is labeled by a variable in V
- Each leaf is labeled by either a variable, a terminal or ϵ; a leaf labeled by ϵ must be the only child of its parent.
- If an interior node labeled by A with children labeled by $X_1, X_2, \ldots X_k$ (from the left), then $A \to X_1 X_2 \cdots X_k$ must be a rule.



Example Parse Tree with yield 011110

Yield of a parse tree is the concatenation of leaf labels (left–right)

Parse Trees and Derivations

Proposition

Let $G = (V, \Sigma, R, S)$ be a CFG. For any $A \in V$ and $\alpha \in (V \cup \Sigma)^*$, $A \stackrel{*}{\Rightarrow} \alpha$ iff there is a parse tree with root labeled A and whose yield is α.

Parse Trees and Derivations

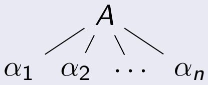
Proposition

Let $G = (V, \Sigma, R, S)$ be a CFG. For any $A \in V$ and $\alpha \in (V \cup \Sigma)^*$, $A \overset{*}{\Rightarrow} \alpha$ iff there is a parse tree with root labeled A and whose yield is α.

Proof.

($\Rightarrow$): Proof by induction on the number of steps in the derivation.

- Base Case: If $A \Rightarrow \alpha$ then $A \to \alpha$ is a rule in G. There is a tree of height 1, with root A and leaves the symbols in α. $\cdots\!\!\longrightarrow$



Parse Tree for Base Case

Parse Trees for Derivations

Proof (contd).

(⇒): Proof by induction on the number of steps in the derivation.

- Induction Step: Let $A \stackrel{*}{\Rightarrow} \alpha$ in $k+1$ steps.

Parse Trees for Derivations

Proof (contd).

(⇒): Proof by induction on the number of steps in the derivation.

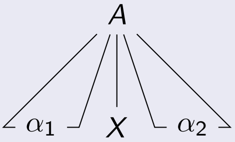
- Induction Step: Let $A \overset{*}{\Rightarrow} \alpha$ in $k+1$ steps.
- Then $A \overset{*}{\Rightarrow} \alpha_1 X \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2 = \alpha$, where $X \to X_1 \cdots X_n = \gamma$ is a rule

Parse Trees for Derivations

Proof (contd).

(⇒): Proof by induction on the number of steps in the derivation.

- Induction Step: Let $A \stackrel{*}{\Rightarrow} \alpha$ in $k+1$ steps.
- Then $A \stackrel{*}{\Rightarrow} \alpha_1 X \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2 = \alpha$, where $X \to X_1 \cdots X_n = \gamma$ is a rule
- By ind. hyp., there is a tree with root A and yield $\alpha_1 X \alpha_2$.



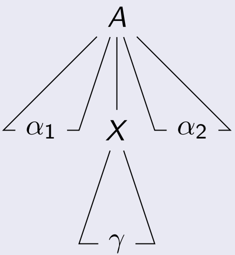
Parse Tree for Induction Step

Parse Trees for Derivations

Proof (contd).

(⇒): Proof by induction on the number of steps in the derivation.

- Induction Step: Let $A \overset{*}{\Rightarrow} \alpha$ in $k+1$ steps.
- Then $A \overset{*}{\Rightarrow} \alpha_1 X \alpha_2 \Rightarrow \alpha_1 \gamma \alpha_2 = \alpha$, where $X \rightarrow X_1 \cdots X_n = \gamma$ is a rule
- By ind. hyp., there is a tree with root A and yield $\alpha_1 X \alpha_2$.
- Add leaves $X_1, \ldots X_n$ and make them children of X. New tree is a parse tree with desired yield. $\cdots\rightarrow$



Parse Tree for Induction Step

Derivations for Parse Trees

Proof (contd).

($\Leftarrow$): Assume that there is a parse tree with root A and yield α. Need to show that $A \stackrel{*}{\Rightarrow} \alpha$.

Derivations for Parse Trees

Proof (contd).

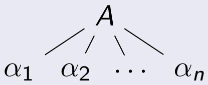
(⇐): Assume that there is a parse tree with root A and yield α. Need to show that $A \stackrel{*}{\Rightarrow} \alpha$. Proof by induction on the number of internal nodes in the tree.

Derivations for Parse Trees

Proof (contd).

(⇐): Assume that there is a parse tree with root A and yield α. Need to show that $A \stackrel{*}{\Rightarrow} \alpha$. Proof by induction on the number of internal nodes in the tree.

- Base Case: If tree has only one internal node, then it has the form as in picture



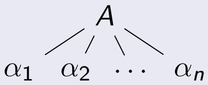
Parse Tree with one internal node

Derivations for Parse Trees

Proof (contd).

($\Leftarrow$): Assume that there is a parse tree with root A and yield α. Need to show that $A \stackrel{*}{\Rightarrow} \alpha$. Proof by induction on the number of internal nodes in the tree.

- Base Case: If tree has only one internal node, then it has the form as in picture
- Then, $\alpha = X_1 \cdots X_n$ and $A \rightarrow \alpha$ is a rule. Thus, $A \stackrel{*}{\Rightarrow} \alpha$.

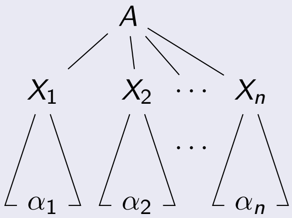


Parse Tree with one internal node

Derivations for Parse Trees

Proof (contd).

(⇐) Induction Step: Suppose α is the yield of a tree with $k+1$ interior nodes. Let $X_1, X_2, \ldots X_n$ be the children of the root ordered from the left. Not all X_i are leaves, and $A \rightarrow X_1 X_2 \cdots X_n$ must be a rule.



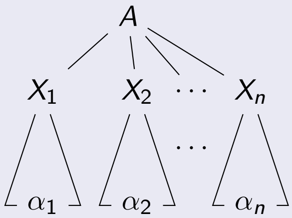
Tree with $k+1$ internal nodes

Derivations for Parse Trees

Proof (contd).

($\Leftarrow$) Induction Step: Suppose α is the yield of a tree with $k+1$ interior nodes. Let $X_1, X_2, \ldots X_n$ be the children of the root ordered from the left. Not all X_i are leaves, and $A \rightarrow X_1 X_2 \cdots X_n$ must be a rule.

- Let α_i be the yield of the tree rooted at X_i; so X_i is a leaf $\alpha_i = X_i$



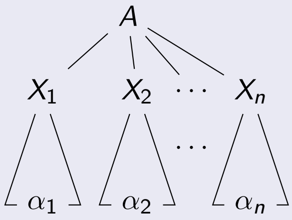
Tree with $k+1$ internal nodes

Derivations for Parse Trees

Proof (contd).

(⇐) Induction Step: Suppose α is the yield of a tree with $k+1$ interior nodes. Let $X_1, X_2, \ldots X_n$ be the children of the root ordered from the left. Not all X_i are leaves, and $A \to X_1 X_2 \cdots X_n$ must be a rule.

- Let α_i be the yield of the tree rooted at X_i; so X_i is a leaf $\alpha_i = X_i$
- Now if $j < i$ then all the descendents of X_j are to the left of the descendents of X_i. So $\alpha = \alpha_1 \alpha_2 \cdots \alpha_n$. $\cdots\!\longrightarrow$

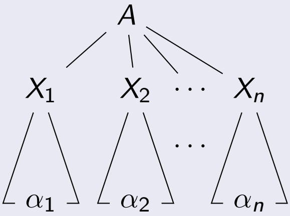


Tree with $k+1$ internal nodes

Derivations for Parse Trees

Proof (contd).

($\Leftarrow$) Induction Step: Suppose α is the yield of a tree with $k+1$ interior nodes.

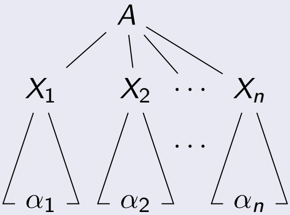


Derivations for Parse Trees

Proof (contd).

(⇐) Induction Step: Suppose α is the yield of a tree with $k+1$ interior nodes.

- Each subtree rooted at X_i has at most k internal nodes. So if X_i is a leaf $X_i \stackrel{*}{\Rightarrow} \alpha_i$ and if X_i is not a leaf then $X_i \stackrel{*}{\Rightarrow} \alpha_i$ (ind. hyp.).

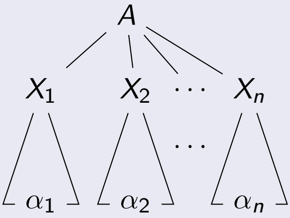


Derivations for Parse Trees

Proof (contd).

($\Leftarrow$) Induction Step: Suppose α is the yield of a tree with $k+1$ interior nodes.

- Each subtree rooted at X_i has at most k internal nodes. So if X_i is a leaf $X_i \stackrel{*}{\Rightarrow} \alpha_i$ and if X_i is not a leaf then $X_i \stackrel{*}{\Rightarrow} \alpha_i$ (ind. hyp.).
- Thus $A \Rightarrow X_1X_2\cdots X_n \stackrel{*}{\Rightarrow} \alpha_1X_2\cdots X_n \stackrel{*}{\Rightarrow} \alpha_1\alpha_2\cdots X_n \stackrel{*}{\Rightarrow} \alpha_1\cdots\alpha_n = \alpha$ □



Recap . . .

For a CFG G with variable A the following are equivalent

1. $A \overset{*}{\Rightarrow} w$
2. There is a parse tree with root A and yield w

Recap . . .

For a CFG G with variable A the following are equivalent

1. $A \stackrel{*}{\Rightarrow} w$
2. There is a parse tree with root A and yield w

Context-free-ness

CFGs have the property that if $X \stackrel{*}{\Rightarrow} \gamma$ then $\alpha X \beta \stackrel{*}{\Rightarrow} \alpha\gamma\beta$

Example: English Sentences

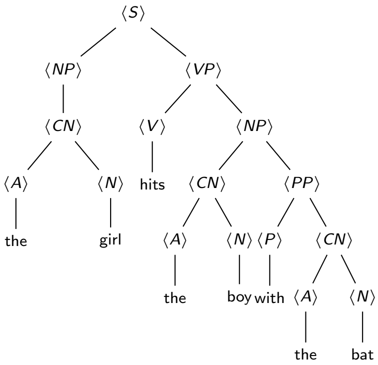
English sentences can be described as

$$
\begin{aligned}
\langle S\rangle &\rightarrow \langle NP\rangle\langle VP\rangle\\
\langle NP\rangle &\rightarrow \langle CN\rangle \mid \langle CN\rangle\langle PP\rangle\\
\langle VP\rangle &\rightarrow \langle CV\rangle \mid \langle CV\rangle\langle PP\rangle\\
\langle PP\rangle &\rightarrow \langle P\rangle\langle CN\rangle\\
\langle CN\rangle &\rightarrow \langle A\rangle\langle N\rangle\\
\langle CV\rangle &\rightarrow \langle V\rangle \mid \langle V\rangle\langle NP\rangle\\
\langle A\rangle &\rightarrow \text{a} \mid \text{the}\\
\langle N\rangle &\rightarrow \text{boy} \mid \text{girl} \mid \text{bat}\\
\langle V\rangle &\rightarrow \text{hits} \mid \text{likes} \mid \text{sees}\\
\langle P\rangle &\rightarrow \text{with}
\end{aligned}
$$

Multiple Parse Trees

Example 1

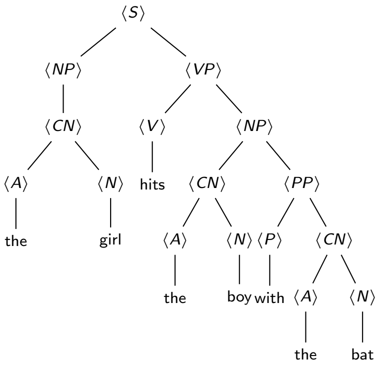
The sentence "the girl hits the boy with the bat" has the following parse tree



Multiple Parse Trees

Example 1

The sentence “the girl hits the boy with the bat” has the following parse trees



Example: Arithmetic Expressions

Consider the language of all arithmetic expressions (E) built out of integers (N) and identifiers (I), using only $+$ and $*$

Example: Arithmetic Expressions

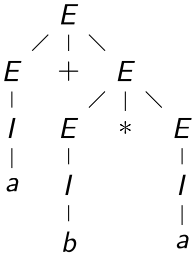
Consider the language of all arithmetic expressions (E) built out of integers (N) and identifiers (I), using only $+$ and $*$
$G_{\rm exp} = (\{E, I, N\}, \{a, b, 0, 1, (,), +, *, -\}, R, E)$ where R is

$$
\begin{aligned}
E &\rightarrow I \mid N \mid -N \mid E+E \mid E*E \mid (E) \\
I &\rightarrow a \mid b \mid Ia \mid Ib \\
N &\rightarrow 0 \mid 1 \mid N0 \mid N1
\end{aligned}
$$

Multiple Parse Trees

Example 2

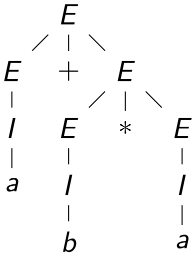
The parse tree for expression $a + b * a$ in the grammar G_{exp} is

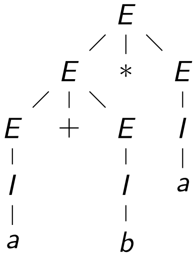


Multiple Parse Trees

Example 2

The parse trees for expression $a + b * a$ in the grammar $G_{\rm exp}$ is





Ambiguity

Definition

A grammar $G = (V, \Sigma, R, S)$ is said to be ambiguous if there is $w \in \Sigma^*$ for which there are two different parse trees.

Ambiguity

Definition

A grammar $G = (V, \Sigma, R, S)$ is said to be ambiguous if there is $w \in \Sigma^*$ for which there are two different parse trees.

Warning!

Existence of two derivations for a string does not mean the grammar is ambiguous!

Removing Ambiguity

Ambiguity maybe removed either by

Removing Ambiguity

Ambiguity maybe removed either by

- Using the semantics to change the rules.

Removing Ambiguity

Ambiguity maybe removed either by

- Using the semantics to change the rules. For example, if we knew who had the bat (the girl or the boy) from the context, we would know which is the right interpretation.

Removing Ambiguity

Ambiguity maybe removed either by

- Using the semantics to change the rules. For example, if we knew who had the bat (the girl or the boy) from the context, we would know which is the right interpretation.
- Adding precedence to operators.

Removing Ambiguity

Ambiguity maybe removed either by

- Using the semantics to change the rules. For example, if we knew who had the bat (the girl or the boy) from the context, we would know which is the right interpretation.
- Adding precedence to operators. For example, $*$ binds more tightly than $+$, or "else" binds with the innermost "if".

An Example

Recall, G_{exp} has the following rules

$$E \rightarrow I \mid N \mid -N \mid E + E \mid E * E \mid (E)$$
$$I \rightarrow a \mid b \mid Ia \mid Ib$$
$$N \rightarrow 0 \mid 1 \mid N0 \mid N1$$

An Example

Recall, G_{exp} has the following rules

$$E \to I \mid N \mid -N \mid E + E \mid E * E \mid (E)$$
$$I \to a \mid b \mid Ia \mid Ib$$
$$N \to 0 \mid 1 \mid N0 \mid N1$$

New CFG G'_{exp} has the rules

$$I \to a \mid b \mid Ia \mid Ib$$
$$N \to 0 \mid 1 \mid N0 \mid N1$$
$$F \to I \mid N \mid -N \mid (E)$$
$$T \to F \mid T * F$$
$$E \to T \mid E + T$$

Ambiguity: Computational Problems

Removing Ambiguity

Problem: Given CFG G, find CFG G' such that $L(G) = L(G')$ and G' is unambiguous.

Ambiguity: Computational Problems

Removing Ambiguity

Problem: Given CFG G, find CFG G' such that $L(G) = L(G')$ and G' is unambiguous.

There is no algorithm that can solve the above problem!

Ambiguity: Computational Problems

Removing Ambiguity

Problem: Given CFG G, find CFG G' such that $L(G) = L(G')$ and G' is unambiguous.
There is no algorithm that can solve the above problem!

Deciding Ambiguity

Problem: Given CFG G, determine if G is ambiguous.

Ambiguity: Computational Problems

Removing Ambiguity

Problem: Given CFG G, find CFG G' such that $L(G) = L(G')$ and G' is unambiguous.
There is no algorithm that can solve the above problem!

Deciding Ambiguity

Problem: Given CFG G, determine if G is ambiguous.
The problem is undecidable.

Problem: Is it the case that for every CFG G, there is a grammar G' such that $L(G) = L(G')$ and G' is unambiguous, *even if G' cannot be constructed algorithmically*?

Inherently Ambiguous Languages

Problem: Is it the case that for every CFG G, there is a grammar G' such that $L(G) = L(G')$ and G' is unambiguous, *even if G' cannot be constructed algorithmically*?
No! There are context-free languages L such that every grammar for L is ambiguous.

Inherently Ambiguous Languages

Problem: Is it the case that for every CFG G, there is a grammar G' such that $L(G) = L(G')$ and G' is unambiguous, *even if G' cannot be constructed algorithmically*?
No! There are context-free languages L such that every grammar for L is ambiguous.

Definition

A context-free language L is said to be inherently ambiguous if every grammar G for L is ambiguous.

Inherently Ambiguous Languages

An Example

Consider

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

Inherently Ambiguous Languages

An Example

Consider

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

One can show that any CFG G for L will have two parse trees on $a^n b^n c^n$, for all but finitely many values of n

Inherently Ambiguous Languages

An Example

Consider

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

One can show that any CFG G for L will have two parse trees on $a^n b^n c^n$, for all but finitely many values of n

- One that checks that number of a's = number of b's

Inherently Ambiguous Languages

An Example

Consider

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k\}$$

One can show that any CFG G for L will have two parse trees on $a^n b^n c^n$, for all but finitely many values of n

- One that checks that number of a's = number of b's
- Another that checks that number of b's = number of c's