Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

# Automata, Languages and Computation

## Chapter 5 : Context-Free Grammars and Languages

Master Degree in Computer Engineering
University of Padua
Lecturer : Giorgio Satta

Lecture based on material originally developed by :
Gösta Grahne, Concordia University

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

# Derivation trees

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

1. Context-free grammars : we consider devices defining structures more complex than regular languages

2. Parse trees : tree representation of a derivation

3. CFGs and ambiguity : some strings might have more than one parse tree

4. Relation with regular languages : CFGs can simulate FAs or regular expressions

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Informal example of CFL

Let $L_{pal} = \{w \mid w \in \Sigma^*, \ w = w^R\}$, also called the language of all **palindrome** strings

**Example** : (ignore case, spaces, and punctuation characters)
"Madam I'm Adam" is a palindrome;
"A man, a plan, a canal, Panama!" is a palindrome

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Informal example of CFL

Let $\Sigma = \{0, 1\}$ and assume $L_{pal}$ is a regular language

Let $n$ be the constant from the pumping lemma. We pick
$w = 0^n 10^n \in L_{pal}$, $w \geqslant n$

Let $w = xyz$ be such that $y \neq \epsilon$ and $|xy| \leqslant n$

If $k = 0$, $xz \notin L_{pal}$ : the number 0's to the left of 1 is smaller than
the number of 0's to its right

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Informal example of CFL

We **inductively** define $L_{pal}$

**Base** $\epsilon$, 0, and 1 are palindrome strings

**Induction**
If $w$ is a palindrome strings, then $0w0$ and $1w1$ are also palindrome strings

Nothing else is a palindrome string

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## CFG example

CFGs are a formalism for **recursively** defining languages such as $L_{pal}$, using **rewriting rules**

1. $P \to \epsilon$

2. $P \to 0$

3. $P \to 1$

4. $P \to 0P0$

5. $P \to 1P1$

$P$ is a **variable** representing strings of a language. In this grammar $P$ is also the initial symbol

Compare variables with recursive functions in programming languages

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Definition

A **context-free grammar** (CFG for short) is a tuple

$$G = (V, T, P, S)$$

where

- $V$ is a finite set of **variables** (also called **nonterminals**)
- $T$ is a finite set of **terminal symbols**, representing the language alphabet
- $P$ is a finite set of **productions** having the form $A \rightarrow \alpha$, where $A$ (head, or left-hand side) is a variable and $\alpha$ (body or right-hand side) is a string in $(V \cup T)^*$
- $S$ is a variable called **initial symbol**

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

A CFG for palindrome strings is

$$G_{pal} = (\{P\}, \{0, 1\}, A, P)$$

with

$$A = \{P \to \epsilon, P \to 0, P \to 1, P \to 0P0, P \to 1P1\}$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

The language of all regular expressions over the alphabet $\{0, 1\}$ can be defined by the CFG

$$G_{regEx} = (\{E\}, T, P, E)$$

where $T$ is defined as ($\epsilon$ **overloaded** !)

$$\{\varnothing, \ \epsilon, \ \mathbf{0}, \ \mathbf{1}, \ +, \ ., \ ^*, \ (, )\}$$

and $P$ is defined as

$$\begin{aligned} \{E \to \varnothing, \ E \to \epsilon, \ E \to \mathbf{0}, \ E \to \mathbf{1}, \\ E \to E.E, \ E \to E + E, \ E \to E^*, E \to (E)\} \end{aligned}$$

Don't get confused: this defines the syntax of regular expressions, not the generated language

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

Consider a simplified form of the **arithmetic expressions** as used in most common programming languages

$+$ and $*$ are arithmetic operators; operands are **identifiers** generated by the regular expression

$$(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b} + \boldsymbol{0} + \boldsymbol{1})^*$$

We use the CFG
$$G = (\{E, I\}, T, P, E)$$

where

- variabile $E$ represents arithmetic expressions
- variabile $I$ represents identifiers

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

$T$ is defined as

$$\{+, *, (, ), a, b, 0, 1\}$$

$P$ contains the following productions

| | |
|---|---|
| 1. $E \rightarrow I$ | 6. $I \rightarrow b$ |
| 2. $E \rightarrow E + E$ | 7. $I \rightarrow I\,a$ |
| 3. $E \rightarrow E * E$ | 8. $I \rightarrow I\,b$ |
| 4. $E \rightarrow (E)$ | 9. $I \rightarrow I\,0$ |
| 5. $I \rightarrow a$ | 10. $I \rightarrow I\,1$ |

We will later present several examples using this CFG

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Compact notation

Usually, productions with a common head are grouped together

**Example** :   Productions $A \rightarrow \alpha_1$, $A \rightarrow \alpha_2$, ..., $A \rightarrow \alpha_n$ can be written in a more compact notation

$$A \rightarrow \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_n$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Test

Define a CFG for each of the following languages

- $L = \{a^n b^n \mid n \geqslant 1\}$
- $L = \{a^n b^m \mid n \geqslant m \geqslant 1\}$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Derivation

In order to generate strings using a CFG, we define a binary relation $\underset{G}{\Rightarrow}$ over $(V \cup T)^*$, called **rewrites**

Let $G = (V, T, P, S)$ be a CFG, $A \in V$, $\{\alpha, \beta\} \subset (V \cup T)^*$. If $A \rightarrow \gamma \in P$ then

$$\alpha A \beta \underset{G}{\Rightarrow} \alpha \gamma \beta$$

and we say that $\alpha A \beta$ **derives in one step** $\alpha \gamma \beta$

If $G$ is understood from the context, we use the simplified notation

$$\alpha A \beta \Rightarrow \alpha \gamma \beta$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Derivation

We define $\stackrel{*}{\Rightarrow}$ as the reflexive and transitive closure of $\Rightarrow$

**Base** Let $\alpha \in (V \cup T)^*$. Then $\alpha \stackrel{*}{\Rightarrow} \alpha$

**Induction** If $\alpha \stackrel{*}{\Rightarrow} \beta$ and $\beta \Rightarrow \gamma$, then $\alpha \stackrel{*}{\Rightarrow} \gamma$

Relation $\stackrel{*}{\Rightarrow}$ is called **derivation**

We often write derivations by indicating all of the **intermediate steps**

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

A possible derivation of $a * (a + b00)$ from $E$ in the CFG for arithmetic expressions :

$$
\begin{aligned}
E &\Rightarrow E * E & &\Rightarrow a * (E + I0) \\
&\Rightarrow E * (E) & &\Rightarrow a * (E + I00) \\
&\Rightarrow I * (E) & &\Rightarrow a * (E + b00) \\
&\Rightarrow a * (E) & &\Rightarrow a * (I + b00) \\
&\Rightarrow a * (E + E) & &\Rightarrow a * (a + b00) \\
&\Rightarrow a * (E + I)
\end{aligned}
$$

Contrast with regular expressions, which do not have derivations for individual strings

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

At each step in a derivation there might be several variables to which we can apply the rewrite relation :

$$I * E \Rightarrow a * E \Rightarrow a * (E)$$
$$I * E \Rightarrow I * (E) \Rightarrow a * (E)$$

Not all choices lead to a derivation of the desired string :

$$I * E \Rightarrow a * E \Rightarrow a * E + E$$

does not lead to a derivation of $a * (a + b00)$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Leftmost derivation

In derivations, we can avoid the choice of variables to be rewritten if we stick to some **canonical** derivation form

The relation $\underset{lm}{\Rightarrow}$ always rewrites the leftmost variable with some production

We also use the reflexive and transitive closure of $\underset{lm}{\Rightarrow}$, written $\underset{lm}{\overset{*}{\Rightarrow}}$, and call it **leftmost derivation**

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

Leftmost derivation of $a * (a + b00)$ :

$$E \underset{lm}{\Rightarrow} E * E \underset{lm}{\Rightarrow} I * E \underset{lm}{\Rightarrow} a * E \underset{lm}{\Rightarrow} a * (E) \underset{lm}{\Rightarrow} a * (E + E)$$

$$\underset{lm}{\Rightarrow} a * (I + E) \underset{lm}{\Rightarrow} a * (a + E) \underset{lm}{\Rightarrow} a * (a + I) \underset{lm}{\Rightarrow} a * (a + I0)$$

$$\underset{lm}{\Rightarrow} a * (a + I00) \underset{lm}{\Rightarrow} a * (a + b00)$$

We conclude that $E \underset{lm}{\overset{*}{\Rightarrow}} a * (a + b00)$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Rightmost derivation

The relation $\underset{rm}{\Rightarrow}$ always rewrites the rightmost variable with the body of a production

We use the reflexive and transitive closure of $\underset{rm}{\Rightarrow}$, written $\underset{rm}{\overset{*}{\Rightarrow}}$, called **rightmost derivation**

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

Rightmost derivation :

$$E \underset{rm}{\Rightarrow} E * E \underset{rm}{\Rightarrow} E * (E) \underset{rm}{\Rightarrow} E * (E + E) \underset{rm}{\Rightarrow} E * (E + I)$$
$$\underset{rm}{\Rightarrow} E * (E + I0) \underset{rm}{\Rightarrow} E * (E + I00) \underset{rm}{\Rightarrow} E * (E + b00)$$
$$\underset{rm}{\Rightarrow} E * (I + b00) \underset{rm}{\Rightarrow} E * (a + b00) \underset{rm}{\Rightarrow} I * (a + b00)$$
$$\underset{rm}{\Rightarrow} a * (a + b00)$$

We conclude that $E \underset{rm}{\overset{*}{\Rightarrow}} a * (a + b00)$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Notation for CFGs

We use the following conventions

- $a, b, c, \ldots$ terminal symbols
- $A, B, C, \ldots$ variables (nonterminal symbols)
- $u, v, w, x, y, z$ terminal strings
- $X, Y, Z$ terminal or nonterminal symbols
- $\alpha, \beta, \gamma, \ldots$ strings over terminal or nonterminal symbols

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Language generated by a CFG

Let $G = (V, T, P, S)$ be some CFG. The **generated language** of $G$ is

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

that is, the set of all strings in $T^*$ that can be derived from the start symbol

$L(G)$ is a **context-free language**, or CFL for short

**Example** :  $L(G_{pal})$ is a CFL

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Test

Consider the language $L$ of all strings over "(" and ")" where parentheses are always **well balanced** (assume $\epsilon \notin L$)

- for the following CFG

$$G \;=\; (\{S\}, \{(,)\}, P, S)$$

  specify the set $P$ such that $L(G) = L$
- produce a derivation for string

$$w \;=\; (\,(\,)\,(\,(\,)\,)\,)$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Language generated by a CFG

$G_{pal} = (\{P\}, \{0, 1\}, A, P)$, where

$$A = \{P \rightarrow \epsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1\}$$

**Theorem** $L(G_{pal}) = \{w \mid w \in \{0, 1\}^*, \ w = w^R\}$

**Proof** ($\supseteq$ part)   Assume $w = w^R$. Using induction on $|w|$, we show $w \in L(G_{pal})$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Language generated by a CFG

**Base** $|w| = 0$ or $|w| = 1$. Then $w$ is $\epsilon, 0$, or else 1. Since $P \to \epsilon$, $P \to 0$, and $P \to 1$ are productions of the grammar, we conclude that $P \underset{G}{\overset{*}{\Rightarrow}} w$

**Induction** Assume now $|w| \geqslant 2$. Since $w = w^R$, we must have $w = 0x0$ or else $w = 1x1$, with $x = x^R$. From the inductive hypothesis we then have $P \overset{*}{\Rightarrow} x$.

If $w = 0x0$, we can write

$$P \Rightarrow 0P0 \overset{*}{\Rightarrow} 0x0 = w$$

Therefore $w \in L(G_{pal})$

Case $w = 1x1$ can be dealt with similarly

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Language generated by a CFG

($\subseteq$ part)    Assume now $w \in L(G_{pal})$. We show $w = w^R$

Since $w \in L(G_{pal})$, we have $P \overset{*}{\Rightarrow} w$. We use induction on the number of steps of the derivation

**Base** The derivation $P \overset{*}{\Rightarrow} w$ has 1 step. Then $w$ must be $\epsilon$, 0, or 1. All the three generated strings are palindrome

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Language generated by a CFG

**Induction** Let $n \geqslant 2$ be the number of steps in the derivation. At the first step only two cases are possible :

$$P \Rightarrow 0P0 \overset{*}{\Rightarrow} 0x0 = w$$

or else

$$P \Rightarrow 1P1 \overset{*}{\Rightarrow} 1x1 = w$$

In both cases, the second part of the derivation implies $P \overset{*}{\Rightarrow} x$ in $n - 1$ steps (this will be explained later in more detail)

By the inductive hypothesis, $x$ is a palindrome string. Then also $w$ is a palindrome string $\qquad\square$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Proofs about CFGs

We need to show that a given CFG generates a desired language

For each variable $A$ in the CFG, define some property $\mathcal{P}_A$ for strings $w$ over the alphabet

Show that, for every $A$, we have

$A \overset{*}{\Rightarrow} w$ *if and only if* $\mathcal{P}_A(w)$ *holds true*

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Proofs about CFGs

If part : if $\mathcal{P}_A(w)$ then $A \overset{*}{\Rightarrow} w$

Use **mutual induction** on $|w|$

- using $\mathcal{P}_A$ definition, choose a factorization $w = x_1 x_2 \cdots x_k$ such that $\mathcal{P}_{B_i}(x_i)$ holds for each $i$
- use the inductive hypothesis on $\mathcal{P}_{B_i}(x_i)$ to obtain $B_i \overset{*}{\Rightarrow} x_i$, for each $i$
- choose a production $A \rightarrow B_1 B_2 \cdots B_k$ and obtain

$$A \Rightarrow B_1 B_2 \cdots B_k$$
$$\overset{*}{\Rightarrow} x_1 B_2 \cdots B_k$$
$$\vdots$$
$$\overset{*}{\Rightarrow} x_1 x_2 \cdots x_k = w$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Proofs about CFGs

Only if part : if $A \overset{*}{\Rightarrow} w$ then $\mathcal{P}_A(w)$ holds true

Use **mutual induction** on the length of derivation $A \overset{*}{\Rightarrow} w$

- focus on the first production of the derivation

$$A \Rightarrow B_1 B_2 \cdots B_k$$
$$\overset{*}{\Rightarrow} x_1 B_2 \cdots B_k$$
$$\vdots$$
$$\overset{*}{\Rightarrow} x_1 x_2 \cdots x_k = w$$

- use the inductive hypothesis on $B_i \overset{*}{\Rightarrow} x_i$ to obtain that $\mathcal{P}_{B_i}(x_i)$ holds, for each $i$
- use $\mathcal{P}_A$ definition to show that $\mathcal{P}_A(w)$ holds true

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Sentential form

Let $G = (V, T, P, S)$ be a CFG and let $\alpha \in (V \cup T)^*$

- if $S \overset{*}{\Rightarrow} \alpha$ we say that $\alpha$ is a **sentential form**

- if $S \overset{*}{\underset{lm}{\Rightarrow}} \alpha$ we say that $\alpha$ is a **left sentential form**

- if $S \overset{*}{\underset{rm}{\Rightarrow}} \alpha$ we say that $\alpha$ is a **right sentential form**

**Note** : $L(G)$ contains the sentential forms in $T^*$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Examples

Consider previous CFG $G$ for a fragment of arithmetic expressions.
Then $E * (I + E)$ is a sentential form, since

$$E \Rightarrow E * E \Rightarrow E * (E) \Rightarrow E * (E + E) \Rightarrow E * (I + E)$$

This derivation is neither leftmost nor rightmost

$a * E$ is a leftmost sentential form, since

$$E \underset{lm}{\Rightarrow} E * E \underset{lm}{\Rightarrow} I * E \underset{lm}{\Rightarrow} a * E$$

$E * (E + E)$ is a rightmost sentential form, since

$$E \underset{rm}{\Rightarrow} E * E \underset{rm}{\Rightarrow} E * (E) \underset{rm}{\Rightarrow} E * (E + E)$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Test

Define a CFG for each of the following languages, describing for each variable the set of generated strings

- $L = \{w \mid w = x2x^R,\ x \in \{0,1\}^*\}$
- $L = \{w \mid w = a^i b^j c^k,\ i,j,k \geqslant 1,\ j \neq k\}$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Test

Describe in words the language generated by the following CFG

$$G = (\{S, Z\}, \{0, 1\}, P, S)$$

where

$$P = \{S \to 0S1 \mid 0Z1, \ Z \to 0Z \mid \epsilon\}$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Derivation composition

We can always compose two derivations $A \stackrel{*}{\Rightarrow} \alpha B \beta$ and $B \stackrel{*}{\Rightarrow} \gamma$ into a single derivation

$$A \stackrel{*}{\Rightarrow} \alpha B \beta \stackrel{*}{\Rightarrow} \alpha \gamma \beta$$

This follows from the hypothesis about rewriting being **independent** from the context (context-free)

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

Consider our CFG for generating arithmetic expressions. Starting
with

$$E \Rightarrow E + E \Rightarrow E + (E)$$
$$E \Rightarrow I \Rightarrow Ib \Rightarrow ab$$

we can compose at the rightmost occurrence of $E$, obtaining

$$E \Rightarrow E + E \Rightarrow E + (E) \Rightarrow E + (I) \Rightarrow E + (Ib) \Rightarrow E + (ab)$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Derivation factorization

Assume $A \Rightarrow X_1 X_2 \cdots X_k \overset{*}{\Rightarrow} w$. We can **factorize** $w$ as $w_1 w_2 \cdots w_k$ such that $X_i \overset{*}{\Rightarrow} w_i$, $1 \leqslant i \leqslant k$

As a special case, we can have $X_i = w_i \in \Sigma$

Substring $w_i$ can be identified from derivation $A \overset{*}{\Rightarrow} w$ by considering **only** those derivation steps that rewrite $X_i$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

Consider $E \Rightarrow E * E \stackrel{*}{\Rightarrow} a * b + a$

We have

$$\underbrace{a}_{E} \quad \underbrace{*}_{*} \quad \underbrace{b + a}_{E}$$

and we can write

$$E \stackrel{*}{\Rightarrow} a$$

$$* \stackrel{*}{\Rightarrow} *$$

$$E \stackrel{*}{\Rightarrow} b + a$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Parse trees

**Parse trees** are a graphical representation alternative to derivations

Intuitively, parse trees represent the **syntactic structure** of a sentence according to the grammar

In compilers, parse trees are the structure of choice when **translating** into executable code

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Parse trees

Let $G = (V, T, P, S)$ be a CFG. An ordered tree is a **parse tree** of $G$ if :

- each internal node is labeled with a variable in $V$
- each leaf node is labeled with a symbol in $V \cup T \cup \{\epsilon\}$; each leaf labeled with $\epsilon$ is the only child of its parent
- if an internal node is labeled $A$ and its children (from left to right) are labeled

$$X_1, X_2, \ldots, X_k$$

then $A \rightarrow X_1 X_2 \cdots X_k \in P$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

CFG for arithmetic expressions and parse tree associated with the derivation $E \Rightarrow E + E \Rightarrow I + E$

1. $E \rightarrow I$
2. $E \rightarrow E + E$
3. $E \rightarrow E * E$
4. $E \rightarrow (E)$

$\vdots$

Context-free grammars
**Parse trees**
CFGs and ambiguity
Relation with regular languages
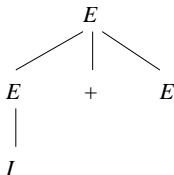
## Example

CFG for palindrome strings and parse tree associated with the derivation $P \Rightarrow 0P0 \Rightarrow 01P10 \Rightarrow 0110$

1. $P \rightarrow \epsilon$
2. $P \rightarrow 0$
3. $P \rightarrow 1$
4. $P \rightarrow 0P0$
5. $P \rightarrow 1P1$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages
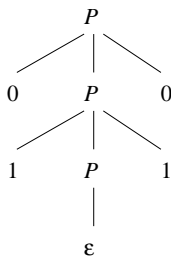
## Parse tree terminology

We use the following terms associated with parse trees

- node and arc
- parent node and child node
- ancestor node and descendant node
- root node, inner node (including the root) and leaf node

**Recall** : For each internal node, the child nodes are **ordered**

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Yeld of a parse tree

The **yield** of a parse tree is the string obtained by reading the
leaves from left to right

Of special importance are the **complete** parse trees, where :

- the yield is a string of terminal symbols
- the root is labeled by the initial symbol

The set of yields of all complete parse trees is the language
generated by the CFG

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example
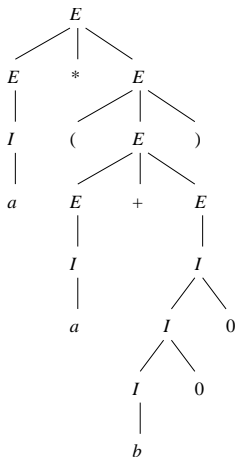


Complete parse tree. The yield is $a * (a + b00)$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Derivations and parse trees

Let $G = (V, T, P, S)$ be a CFG, $A \in V$ and $w \in T^*$. The following statements are equivalent (statements must all be true or must all be false) :

- $A \overset{*}{\Rightarrow} w$
- $A \overset{*}{\underset{lm}{\Rightarrow}} w$
- $A \overset{*}{\underset{rm}{\Rightarrow}} w$
- there exists a parse tree for $G$ with root label $A$ and yield $w$

Proof not required for these theorems

Relation between derivations and parse trees **is not** one-to-one (see next slides)

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Derivations and parse trees

A parse tree can be associated with **several** derivations

**Example** :   Consider the CFG with productions $S \to AB$, $A \to a$,
$B \to b$. The parse tree

$$
\begin{array}{c}
S \\
\diagup \quad \diagdown \\
A \qquad B \\
| \qquad | \\
a \qquad b
\end{array}
$$
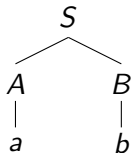
is associated with two derivations

$$S \Rightarrow AB \Rightarrow aB \Rightarrow ab$$

$$S \Rightarrow AB \Rightarrow Ab \Rightarrow ab$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Derivations and parse trees

A derivation can be associated with **several** parse trees

**Example** :   Consider the CFG with productions $S \to SS \mid a$ The derivation

$$S \Rightarrow SS \Rightarrow SSS \Rightarrow aSS \Rightarrow aaS \Rightarrow aaa$$

is associated with two parse trees

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages
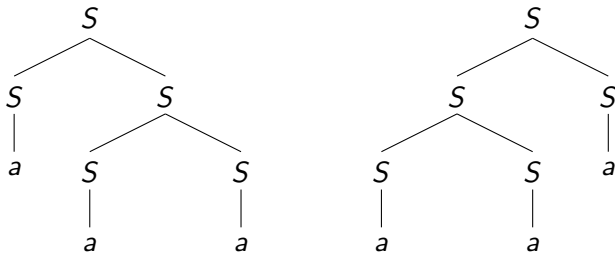
## Ambiguous CFGs

In the CFG

| | | | |
|---|---|---|---|
| 1. $E \rightarrow I$ | | 6. $I \rightarrow b$ | |
| 2. $E \rightarrow E + E$ | | 7. $I \rightarrow I a$ | |
| 3. $E \rightarrow E * E$ | | 8. $I \rightarrow I b$ | |
| 4. $E \rightarrow (E)$ | | 9. $I \rightarrow I 0$ | |
| 5. $I \rightarrow a$ | | 10. $I \rightarrow I 1$ | |

the sentential form $E + E * E$ has two derivations

$$E \Rightarrow E + E \Rightarrow E + E * E$$
$$E \Rightarrow E * E \Rightarrow E + E * E$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Ambiguous CFGs

Associated parse trees for the derivations of $E + E * E$
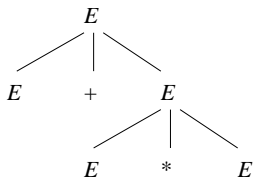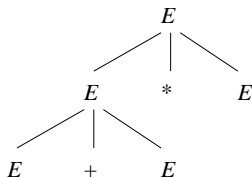


(a)                    (b)

The two derivations correspond to different **precedences** for operators sum and multiplication

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Ambiguous CFGs

The existence of different derivations for a string is not problematic, if these correspond to a single parse tree

**Example** :   In our CFG for arithmetic expressions, the string $a + b$ has at least two derivations

$$E \Rightarrow E + E \Rightarrow I + E \Rightarrow a + E \Rightarrow a + I \Rightarrow a + b$$
$$E \Rightarrow E + E \Rightarrow E + I \Rightarrow I + I \Rightarrow I + b \Rightarrow a + b$$

However, the associated parse trees are the same, and string $a + b$ is **not** ambiguous

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Ambiguous CFGs

Let $G = (V, T, P, S)$ be a CFG. $G$ is **ambiguous** if there exists a string in $L(G)$ with more than one parse tree
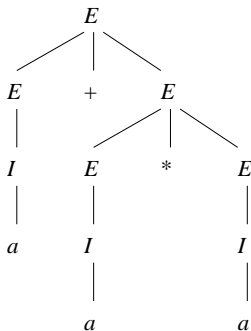
If every string in $L(G)$ has only one parse tree, $G$ is said to be **unambiguous**

The ambiguity is **problematic** in many applications where the syntactic structure of a sentence is used to interpret its meaning
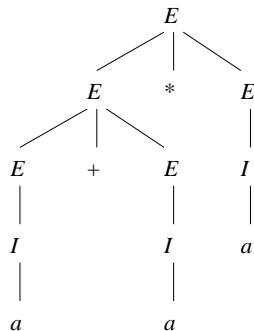
Example: compilers for programming languages

Context-free grammars
Parse trees
**CFGs and ambiguity**
Relation with regular languages

## Example

In the CFG for arithmetic expressions, the terminal string $a + a * a$ has two parse trees



(a)                    (b)

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Canonical derivations

A parse tree is associated with a **unique** leftmost derivation

A leftmost derivation is associated with a **unique** parse tree

More than one leftmost derivations always imply more than one parse trees

Similarly for rightmost derivations

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Inherent ambiguity

A CFL $L$ is **inherently ambiguous** when every CFG such that $L(G) = L$ is ambiguous

**Example** :   Let us consider the language

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

$L$ can be generated by a CFG with the following productions

$$
\begin{aligned}
S &\to AB \mid C \\
A &\to aAb \mid ab \\
B &\to cBd \mid cd \\
C &\to aCd \mid aDd \\
D &\to bDc \mid bc
\end{aligned}
$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Inherent ambiguity

There are two parse trees for the string *aabbccdd*



(a)                                    (b)

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages
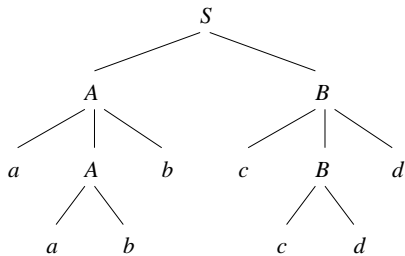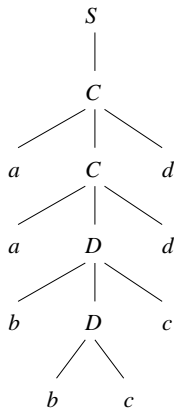
## Inherent ambiguity

Associated leftmost derivations

$$S \underset{lm}{\Rightarrow} AB \underset{lm}{\Rightarrow} aAbB \underset{lm}{\Rightarrow} aabbB \underset{lm}{\Rightarrow} aabbcBd \underset{lm}{\Rightarrow} aabbccdd$$

$$S \underset{lm}{\Rightarrow} C \underset{lm}{\Rightarrow} aCd \underset{lm}{\Rightarrow} aaDdd \underset{lm}{\Rightarrow} aabDcdd \underset{lm}{\Rightarrow} aabbccdd$$

It is possible to show that **every** CFG generating $L$ provides a similar ambiguity for the string *aabbccdd* (not in the textbook)

Language $L$ is therefore inherently ambiguous

Context-free grammars
Parse trees
**CFGs and ambiguity**
Relation with regular languages

## Exercises

- Provide an example showing that the CFG with productions

$$S \rightarrow aS \mid aSbS \mid \epsilon$$

  is ambiguous

- Provide an example showing that the CFG with productions

$$S \rightarrow aSbS \mid bSaS \mid \epsilon$$

  is ambiguous. **Hint**: consider some string of length 4

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

# Reguar languages and CFL

A regular language is **always** a CFL

From a regular expression or from an FA we can aways construct a CFG generating the same language

This is not in the textbook!

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

# From regular expression to CFG

Let $E$ be any regular expression. We use a variable for $E$ (start symbol) and a variable for each subexpression of $E$

We use **structural induction** on the regular expression to build the productions of our CFG

- if $E = a$, then add production $E \rightarrow a$
- if $E = \epsilon$, then add production $E \rightarrow \epsilon$
- if $E = \varnothing$, then production set is empty
- if $E = F + G$, then add production $E \rightarrow F \mid G$
- if $E = FG$, then add production $E \rightarrow FG$
- if $E = F^*$, then add production $E \rightarrow FE \mid \epsilon$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

Regular expression : $\mathbf{0^*1(0+1)^*}$

Use left-associativity for concatenation

CFG :

$$E \rightarrow AR$$
$$R \rightarrow BC$$
$$A \rightarrow 0A \mid \epsilon$$
$$B \rightarrow 1$$
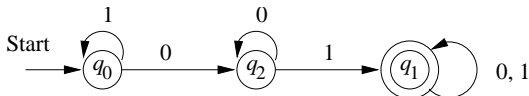$$C \rightarrow DC \mid \epsilon$$
$$D \rightarrow 0 \mid 1$$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

# From FA to CFG

We use a variable $Q$ for each state $q$ of the FA. Initial symbol is $Q_0$

For each transition from state $p$ to state $q$ under symbol $a$, add production $P \rightarrow a\ Q$

If $q$ is a final state, add production $Q \rightarrow \epsilon$

Context-free grammars
Parse trees
CFGs and ambiguity
Relation with regular languages

## Example

Automaton :



CFG :

$$Q_0 \rightarrow 1Q_0 \mid 0Q_2$$
$$Q_2 \rightarrow 0Q_2 \mid 1Q_1$$
$$Q_1 \rightarrow 0Q_1 \mid 1Q_1 \mid \epsilon$$

String 1101 is accepted by the automaton. In the equivalent CFG, 1101 has the following derivation :

$$Q_0 \Rightarrow 1Q_0 \Rightarrow 11Q_0 \Rightarrow 110Q_2 \Rightarrow 1101Q_1 \Rightarrow 1101$$