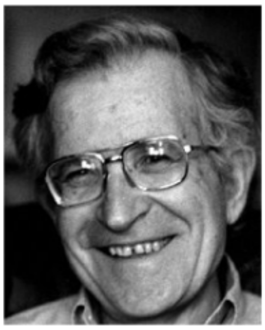
Context Free Language

(Reference:

1. CS311 Computational Structures Context-free Languages: Grammars and Automata Andrew Black Andrew Tolmach)

2. Linz, An Introduction to Formal Languages and Automata, Jones & Bartlett 2000
3. Sipser, Introduction to the Theory of Computation, PWS 1997
4. Sudkamp, Languages and Machines, Addison Wesley 1998
5. Lewis-Papadimitriou, Elements of the Theory of Computation, Prentice Hall 1998)



Chomsky hierarchy

In 1957, Noam Chomsky published *Syntactic Structures*, an landmark book that defined the so-called Chomsky hierarchy of languages

original name	language generated	productions:
Type-3 Grammars	Regular	$A \to \alpha$ and $A \to \alpha B$
Type-2 Grammars	Contex-free	$A \to \gamma$
Type-1 Grammars	Context-sensitive	$\alpha A \beta \to \alpha \gamma \beta$
Type-0 Grammars	Recursively-enumerable	no restriction

A, B: variables, a, b terminals, α, β sequences of terminals and variables

Context-free Grammars

- More general productions than regular grammars

$S \rightarrow w$ where w is any string of terminals and non-terminals

- What languages do these grammars generate?

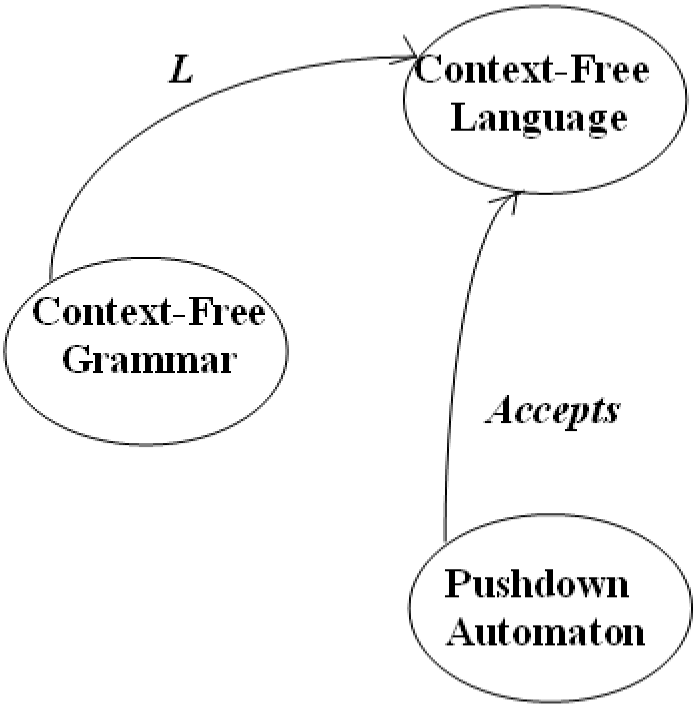
$S \rightarrow (A)$
$A \rightarrow \varepsilon \mid aA \mid ASA$

$S \rightarrow \varepsilon \mid aSb$

Context-free languages more general than regular languages

- $\{a^n b^n \mid n \geq 0\}$ is not regular
  - but it *is* context-free
- Why are they called "context-free"?
  - Context-sensitive grammars allow more than one symbol on the lhs of productions

Context-Free Grammars, Languages, and Pushdown Automata



Context-Free Grammars

Remove all restrictions on the form of the right hand sides.

$S \to abDeFGab$

Keep requirement for single non-terminal on left hand side.

$S \to$

but not $ASB \to$
or $aSb \to$
or $ab \to$

Examples:

Balanced parentheses	$\mathbf{a^n b^n}$
$S \to \varepsilon$	$S \to a\ S\ b$
$S \to SS$	$S \to \varepsilon$
$S \to (S)$	

Context-Free Grammars

A context-free grammar G is a quadruple
(N, T, P, S), where:

- N is the rule alphabet, which contains nonterminals (symbols that are used in the grammar but that do not appear in strings in the language).
- T (the set of terminals)
- P (the set of rules) is a finite subset of $(N - T) \times N^*$,
- S (the start symbol) is an element of N.

$x \Rightarrow_G y$ is a binary relation where $x, y \in N^*$ such that $x = \alpha A \beta$ and $y = \alpha \chi \beta$ for some rule $A \rightarrow \chi$ in P.

Any sequence of the form

$$w_0 \Rightarrow_G w_1 \Rightarrow_G w_2 \Rightarrow_G \ldots \Rightarrow_G w_n$$

e.g., (S) ⇒ (SS) ⇒ ((S)S)

is called a **derivation in G**. Each w_i's is called a **sentinel form**.

The **language generated by G** is

$\{w \in \Sigma^*: S \Rightarrow_G^* w\}$.

A language L is context free if L = L(G) for some context-free grammar G.

AMBIGUOUS GRAMMAR

Ambiguous Grammar:

A context-free grammar G is ambiguous if there is a string $w \in L(G)$ which has:

two different derivation trees

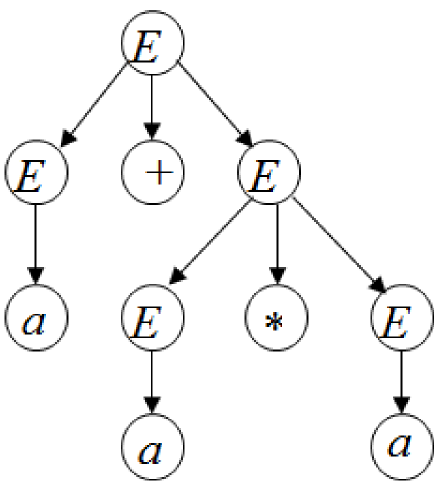
or

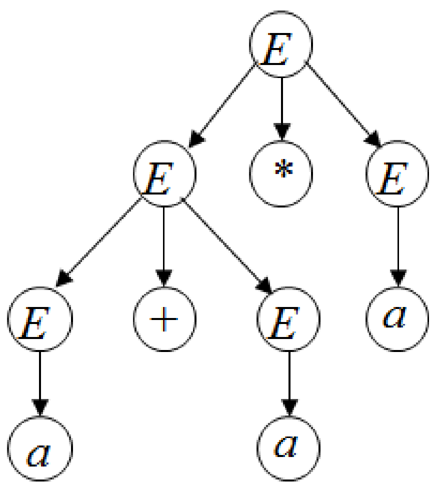
two leftmost derivations

(Two different derivation trees give two different leftmost derivations and vice-versa)

Example: $\boxed{E \to E + E \ \mid \ E * E \ \mid \ (E) \ \mid \ a}$

this grammar is ambiguous since string $a + a * a$ has two derivation trees





$$\boxed{E \rightarrow E + E \quad | \; E * E \; | \; (E) \; | \; a}$$

this grammar is ambiguous also because

string $a + a * a$ has two leftmost derivations

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

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

A successful example:

Ambiguous Grammar

$$E \rightarrow E + E$$
$$E \rightarrow E * E$$
$$E \rightarrow (E)$$
$$E \rightarrow a$$

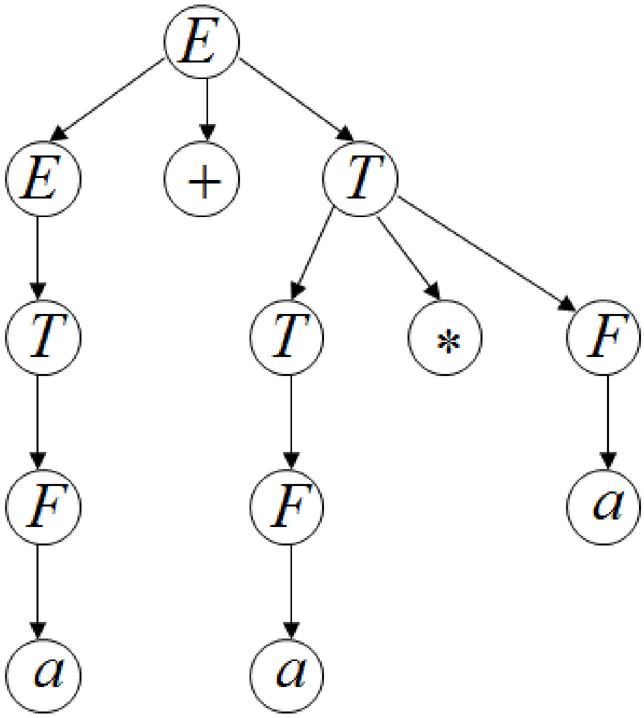
Equivalent Non-Ambiguous Grammar

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid a$$

generates the same language

$$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow a+T \Rightarrow a+T*F$$
$$\Rightarrow a+F*F \Rightarrow a+a*F \Rightarrow a+a*a$$

$$\boxed{\begin{aligned} E &\rightarrow E+T \mid T \\ T &\rightarrow T*F \mid F \\ F &\rightarrow (E) \mid a \end{aligned}}$$



Unique derivation tree for $a+a*a$

Inherently ambiguous grammar:

Inherently Ambiguous Languages

A CFL L is *inherently ambiguous* if *every* CFG for L is ambiguous.

- Such things exist; see course reader.

Example

The language of our example grammar is not inherently ambiguous, even though the grammar is ambiguous.

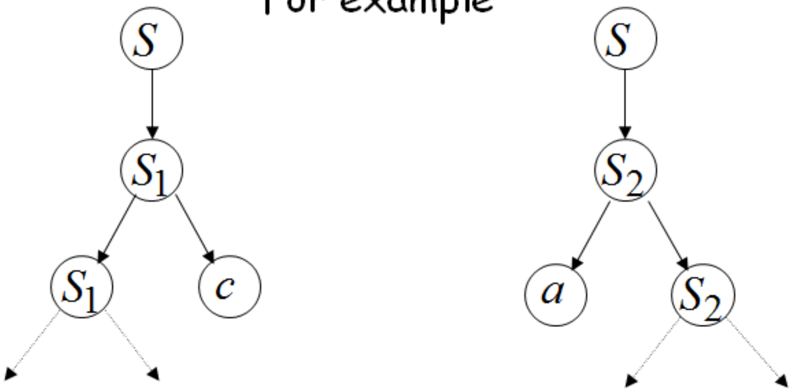
$$\begin{aligned} S &\rightarrow AS \mid \epsilon \\ A &\rightarrow 0A1 \mid B \\ B &\rightarrow B1 \mid 01 \end{aligned}$$

The string $a^n b^n c^n \in L$

has always two different derivation trees

(for any grammar)

For example



Problem 1. (3.2.1)

Which language generates the grammar G given by the productions

$$S \rightarrow aSa \mid aBa$$

$$B \rightarrow bB \mid b$$

Problem 2. (3.2.2)

Find a CFG that generates the language:

$$L(G) = \{ a^n b^m c^m d^{2n} \mid n \geq 0,\ m > 0\}.$$

Problem 3. (3.2.4)

Find a CFG that generates the language

$$L(G) = \{ a^n b^m \mid 0 \leq n \leq m \leq 2n\}.$$

Problem 4. (3.2.5)

Consider the grammar

$$S \rightarrow abScB \mid \lambda$$
$$B \rightarrow bB \mid b$$

Problem 1. **3.2.1**

Which language generates the grammar G given by the productions

$$S \to aSa \mid aBa$$
$$B \to bB \mid b$$

Solution

$L(G) = \{ a^n b^m a^n \mid n > 0, m > 0\}$.

The rule $S \to aSa$ recursively builds an equal number of a*'s* on each end of the string. The recursion is terminated by the application of the rule $S \to aBa$, ensuring at least one leading and one trailing a. The recursive B rule then generates any number of b*'s*. To remove the variable B from the string and obtain a sentence of the language, the rule $B \to b$ must be applied, forcing the presence of at least one b.

Problem 2. **3.2.2**

Find a CFG that generates the language

$L(G) = \{ a^n b^m c^m d^{2n} \mid n \geq 0, m > 0\}$.

Solution

The relationship between the number of leading a*'s* and trailing d*'s* in the language indicates that the recursive rule is needed to generate them. The same is true for b*'s* and c*'s*. Derivations in the grammar

$$S \to aSdd \mid A$$
$$A \to bAc \mid bc$$

generate strings in an outside-to-inside manner. The S rules produce the a's and d's while the A rules generate b's and c's. The rule $A \to bc$, whose application terminates the recursion, ensures the presence of the substring bc in every string in the language.

Problem 3. **3.2.4**

Find a CFG that generates the language

$L(G) = \{ a^n b^m \mid 0 \le n \le m \le 2n\}$.

Solution

$$S \to aSb \mid aSbb \mid \lambda$$

The first recursive rule of G generates a trailing b for every a, while the second generates two b's for each a. Thus there is at least one b for every a and at most two, as specified in the language.

Problem 4. **3.2.5**

Consider the grammar

$$S \to abScB \mid \lambda$$
$$B \to bB \mid b$$

What language does it generate?

Solution

The recursive S rule generates an equal number of ab's and cB's. The B rules generate b^+. In a derivation, each occurrence of B may produce a different number of b's. For example in the derivation

$$\begin{aligned} S &\Rightarrow abScB \\ &\Rightarrow ababScBcB \\ &\Rightarrow ababcBcB \\ &\Rightarrow ababcbcB \\ &\Rightarrow ababcbcbB \\ &\Rightarrow ababcbcbb, \end{aligned}$$

the first occurrence of B generates a single b and the second occurrence produces bb. The language of the grammar consists of the set $L(G) = \{ (ab)^n (cb^m)^n \mid n \ge 0,\ m > 0\}$.

Problem 11. Binary strings with twice as many 1s as 0s.

Solution

$$S \to \lambda \mid 0S1S1S \mid 1S0S1S \mid 1S1S0S$$

Problem 14. Explain why the grammar below is ambiguous.

$S \to 0A \mid 1B$
$A \to 0AA \mid 1S \mid 1$
$B \to 1BB \mid 0S \mid 0$

Solution

The grammar is ambiguous because we can find strings which have multiple derivations:

$$S \Rightarrow 0A \Rightarrow 00AA \Rightarrow 001S1 \Rightarrow 0011B1 \Rightarrow 001101$$
$$S \Rightarrow 0A \Rightarrow 00AA \Rightarrow 0011S \Rightarrow 00110A \Rightarrow 001101$$

Problem 15. Given the following ambiguous context free grammar

$S \to Ab \mid aaB$
$A \to a \mid Aa$
$B \to b$

Solution

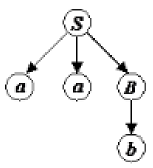
(a) Find the string s generated by the grammar that has two leftmost derivations. Show the derivations.

The string $s = aab$ has the following two leftmost derivations

$$S \Rightarrow aaB \Rightarrow aab$$
$$S \Rightarrow AB \Rightarrow AaB \Rightarrow aaB \Rightarrow aab$$

(b) Show the two derivation trees for the string s.
The two derivation trees of string *aab* are shown below.

$S \Rightarrow aaB \Rightarrow aab$



$S \Rightarrow AB \Rightarrow AaB \Rightarrow aaB \Rightarrow aab$