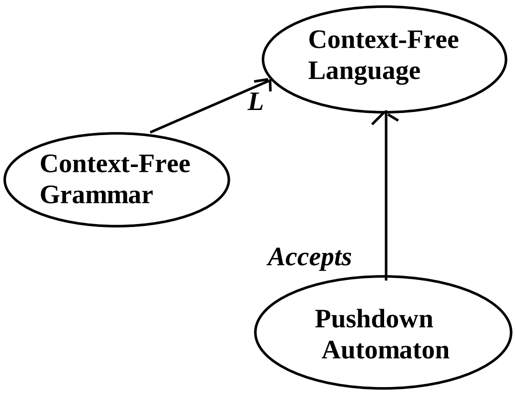
# Context-Free Grammars

Read K & S 3.1
Read Supplementary Materials: Context-Free Languages and Pushdown Automata: Context-Free Grammars
Read Supplementary Materials: Context-Free Languages and Pushdown Automata: Designing Context-Free Grammars.
Do Homework 11.

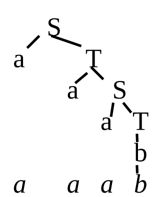
**Context-Free Grammars, Languages, and Pushdown Automata**




**Grammars Define Languages**

Think of grammars as either generators or acceptors.

Example: L = {w ∈ {a, b}* : |w| is even}

| | **Regular Expression** | **Regular Grammar** |
|---|---|---|
| | (aa ∪ ab ∪ ba ∪ bb)* | S → ε<br>S → aT<br>S → bT<br>T → a<br>T → b<br>T → aS<br>T → bS |
| Derivation<br>(Generate) | choose aa<br>choose ab<br>yields<br><br>*a a a b* |  |
| Parse (Accept) | use corresponding FSM | |

## Derivation is Not Necessarily Unique

Example: L = {w ∈ {a, b}* : there is at least one a}

**Regular Expression**

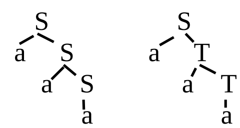
(a ∪ b)*a (a ∪ b)*

choose a from (a ∪ b)
choose a from (a ∪ b)
choose a

choose a
choose a from (a ∪ b)
choose a from (a ∪ b)

**Regular Grammar**

S → a
S → bS
S → aS
S → aT
T → a
T → b
T → aT
T → bT



## More Powerful Grammars

Regular grammars must always produce strings one character at a time, moving left to right.

But sometimes it's more natural to describe generation more flexibly.

Example 1: L = ab*a

| | | |
|---|---|---|
| S → aBa<br>B → ε<br>B → bB | vs. | S → aB<br>B → a<br>B → bB |

Example 2: L = $a^n b^* a^n$

S → B
S → aSa
B → ε
B → bB

Key distinction: Example 1 has no recursion on the nonregular rule.

## Context-Free Grammars

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

S → abDeFGab

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

S →

but not ASB → or aSb → or ab →

Examples:

**balanced parentheses**

S → ε
S → SS
S → (S)

**$a^n b^n$**

S → a S b
S → ε

## Context-Free Grammars

A context-free grammar G is a quadruple (V, Σ, R, S), where:

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

$\mathbf{x \Rightarrow_G y}$ is a binary relation where $x, y \in V^*$ such that $x = \alpha A\beta$ and $y = \alpha\chi\beta$ for some rule $A \rightarrow \chi$ in R.

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 $w_i$ 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.

## Example Derivations

G = (W, Σ, R, S), where

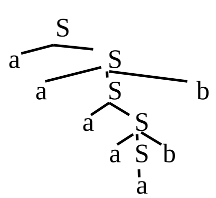
W = {S} ∪ Σ,

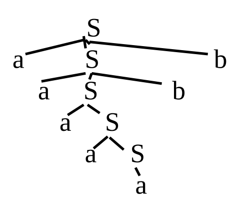
Σ = {a, b},

R = { $S \rightarrow a$,

$S \rightarrow aS$,

$S \rightarrow aSb$ }




## Another Example - Unequal a's and b's

$L = \{a^n b^m : n \neq m\}$

G = (W, Σ, R, S), where

W = {a, b, S, A, B},

Σ = {a, b},

R =

$S \rightarrow A$ /* more a's than b's

$S \rightarrow B$ /* more b's than a's

$A \rightarrow a$

$A \rightarrow aA$

$A \rightarrow aAb$

$B \rightarrow b$

$B \rightarrow Bb$

$B \rightarrow aBb$

## English

S → NP VP
NP → the NP1 | NP1
NP1 → ADJ NP1 | N
ADJ → big | youngest | oldest
N → boy | boys
VP → V | V NP
V → run | runs

the boys run
big boys run
the youngest boy runs

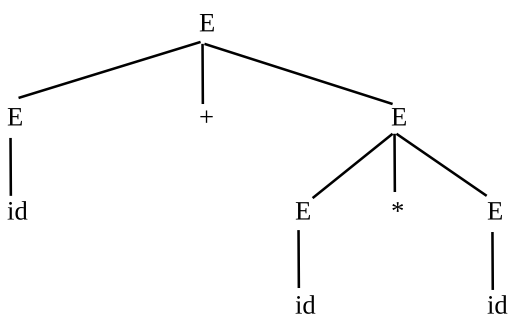
the youngest oldest boy runs
the boy run

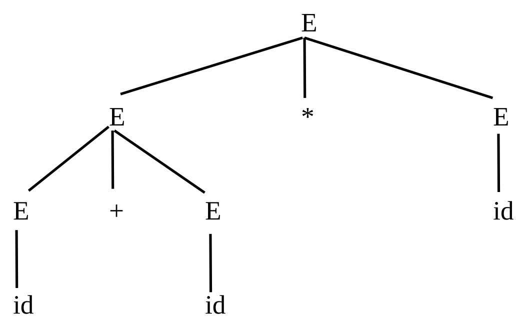
Who did you say Bill saw coming out of the hotel?

## Arithmetic Expressions

The Language of Simple Arithmetic Expressions

G = (V, Σ, R, E), where
V = {+, *, id, T, F, E},
Σ = {+, *, id},
R = { E → id
E → E + E
E → E * E }




id + (id * id)        (id + id) * id

## Arithmetic Expressions -- A Better Way

The Language of Simple Arithmetic Expressions

G = (V, Σ, R, E), where
V = {+, *, (, ), id, T, F, E},
Σ = {+, *, (, ), id},
R = { E → E + T
E → T
T → T * F
T → F
F → (E)
F → id }

Examples:

id + id * id

id * id * id

## BNF

Backus-Naur Form (BNF) is used to define the syntax of programming languages using context-free grammars.

Main idea: give descriptive names to nonterminals and put them in angle brackets.

Example: arithmetic expressions:

⟨expression⟩ → ⟨expression⟩ + ⟨term⟩
⟨expression⟩ → ⟨term⟩
⟨term⟩ → ⟨term⟩ * ⟨factor⟩
⟨term⟩ → ⟨factor⟩
⟨factor⟩ → (⟨expression⟩)
⟨factor⟩ → ⟨id⟩

## The Language of Boolean Logic

$G = (V, \Sigma, R, E)$, where

$V = \{\wedge, \vee, \neg, \Rightarrow, (, ), id, E, E1, E2, E3, E4\}$,
$\Sigma = \{\wedge, \vee, \neg, \Rightarrow, (, ), id\}$,
R = { E → E ⇒ E1
E → E1
E1 → E1 ∨ E2
E1 → E2
E2 → E2 ∧ E3
E2 → E3
E3 → ¬ E4
E3 → E4
E4 → (E)
E4 → id }

## Boolean Logic isn't Regular

Suppose it were regular. Then there is an N as specified in the pumping theorem.

Let w be a string of length 2N + 1 + 2|id| of the form:

$$w = \underbrace{((((((}_{N}\ id\ ))))))\Rightarrow id$$

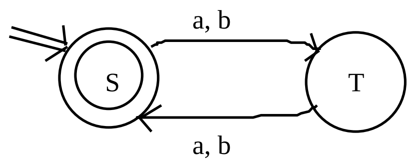
x y

$y = (^k$ for some $k > 0$ because $|xy| \leq N$.

Then the string that is identical to w except that it has k additional ('s at the beginning would also be in the language. But it can't be because the parentheses would be mismatched. So the language is not regular.

**All Regular Languages Are Context Free**

(1) Every regular language can be described by a regular grammar. We know this because we can derive a regular grammar from any FSM (as well as vice versa). Regular grammars are special cases of context-free grammars.




(2) The context-free languages are precisely the languages accepted by NDPDAs. But every FSM is a PDA that doesn't bother with the stack. So every regular language can be accepted by a NDPDA and is thus context-free.

(3) Context-free languages are closed under union, concatenation, and Kleene *, and ε and each single character in Σ are clearly context free.