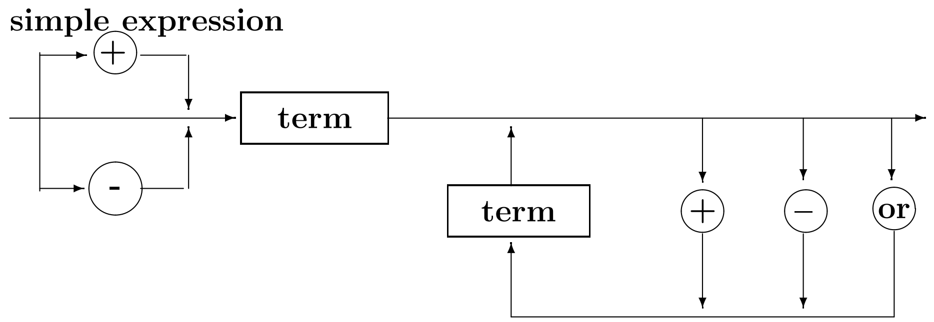
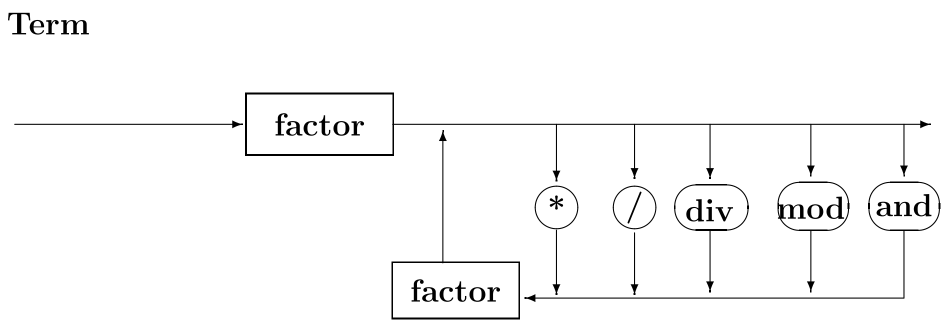
IV. CONTEXT – FREE LANGUAGES

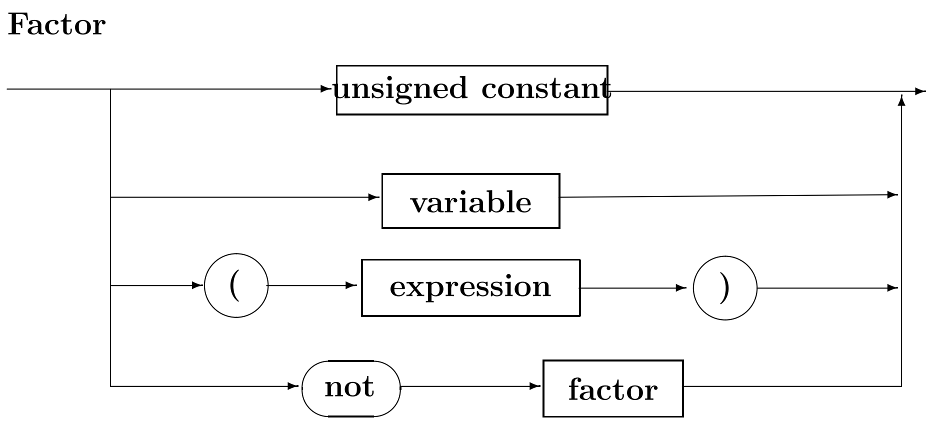
— **The most widely used specification tool for the syntactic structure of programming languages.**

<u>Example</u> Various ways to specify the syntax of a programming language.

(1) <u>Syntax Charts</u>







(2) BNF

<simple expression> ::= <term> | + <term> | - <term> |
<simple expression> + <term> |
<simple expression> - <term> |
<simple expression> **or** <term> |

<term> ::= <factor> | <term> * <factor> |
<term> / <factor> | <term> **div** <factor> |
<term> **mod** <factor> | <term> **and** <factor>

<factor> ::= <unsigned constant> | <variable> |
(<simple expression>) | **not** <factor>

(3) Context-free Grammars

$$
\begin{aligned}
E &\rightarrow T \mid +T \mid -T \mid E+T \mid E-T \mid E \textbf{ or } T \\
T &\rightarrow F \mid T*F \mid T/F \mid T \textbf{ div } F \mid T \textbf{ mod } F \\
&\quad\; \mid T \textbf{ and } F \\
F &\rightarrow a \mid (E) \mid \textbf{not } F
\end{aligned}
$$

1. Context-free grammars

Definition A context-free grammar (CFG) G **is specified by a quadruple** $(N,\ \Sigma,\ P,\ S)$ **where**

N **: the set of nonterminals (variables);**

Σ **: the set of terminals,** $\Sigma \cap N\ =\ \emptyset$**;**

$P \subseteq N \times (N \cup \Sigma)^*$ **: the set of productions;**

$S\ \in\ N$ **: sentence symbol;**

and $N,\ \Sigma,$ **and** P **are all finite.**

Examples

(1) Define a CFG for $\{a^n b^n \mid n \geq 0\}$

$$S \to aSb \mid \varepsilon \qquad\qquad N = \{S\},\ \Sigma = \{a, b\}$$

(2) Define a CFG for $\{a^m b^n \mid m \geq n \geq 0\}$

$$S \to aSb \mid aS \mid \varepsilon$$

Rewriting or derivation

— **A CFG generates a word by rewriting (or derivation)**

— **Let** $G = (N,\ \Sigma,\ P,\ S)$ **be a CFG and** $\beta,\ \beta' \in (N \cup \Sigma)^*$**. If** $\underline{\beta = \beta_1 A \beta_2}$**, for** $A \in N,\ \beta_1,\ \beta_2 \in (N \cup \Sigma)^*$, $\underline{A \to \alpha\ \in P}$**, and** $\underline{\beta' = \beta_1 \alpha \beta_2}$**, then we say that** **β can be rewritten as β'**, **or say that** **β derives β'**, **denoted**

$$\beta \ \Rightarrow \ \beta'$$

$$\text{or } \beta_1 A \beta_2 \ \Rightarrow \ \beta_1 \alpha \beta_2$$

— **So,** $\Rightarrow$ **is a binary relation over** $(N \cup \Sigma)^*$**.**

$\beta \ \Rightarrow^i \ \beta',\ i > 0$, **if** β' **can be obtained from** β **in** i **rewriting steps.**

$\beta \;\Rightarrow^+\; \beta'$ **if** β' **can be obtained from** β **in at least one rewriting step.**
$\beta \;\Rightarrow^*\; \beta'$ **if** $\beta = \beta'$ **or** $\beta \;\Rightarrow^+\; \beta'$**.**

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

Example

$G \;=\; (N,\; \Sigma,\; P,\; S)$ **where**

$$N = \{S\}$$
$$\Sigma = \{a,\; b\}$$
$$P: \quad S \;\rightarrow\; aSbb \mid \varepsilon$$

Then

$$S \Rightarrow \varepsilon$$
$$S \Rightarrow aSbb \Rightarrow abb$$
$$S \Rightarrow aSbb \Rightarrow aaSbbbb \Rightarrow aabbbb$$

For $i > 0$,

$$S \Rightarrow^i a^i S(bb)^i \Rightarrow a^i b^{2i}$$

Intuitively,

$$L(G) = \{a^i b^{2i} \mid i \geq 0\}$$

Example

$G = (N,\ \Sigma,\ P,\ S)$ **where**

$$
\begin{aligned}
N &= \{S,\ A,\ B\}, \quad \Sigma = \{a,\ b\}, \\
P &: \ S \to A \mid B \\
& \quad\ A \to aA \mid \varepsilon \\
& \quad\ B \to aBb \mid ab
\end{aligned}
$$

It is clear that

$$L(G) = \{a^i \mid i \geq 0\} \cup \{a^i b^i \mid i \geq 1\}$$

Example

$G = (N,\ \Sigma,\ P,\ S)$ **where**

$N = \{S\}$

$\Sigma = \{a,\ b\}$

$P \ : \quad S \to \varepsilon \mid aSb \mid bSa \mid SS$

Consider,

$$S \Rightarrow aSb \Rightarrow abSab \Rightarrow abab$$

$$
\begin{aligned}
S &\Rightarrow SS \Rightarrow aSbS \Rightarrow aSbbSa \Rightarrow aaSbbbSa \\
&\Rightarrow aaSbbbbSaa \Rightarrow aabbbbSaa \\
&\Rightarrow aabbbbaa
\end{aligned}
$$

$L(G) = \{w \in \{a,\ b\}^* \mid |w|_a = |w|_b\}$, **intuitively.**

Formally, we prove it as follows:

Given

(1) $G = (\{S\},\ \{a, b\},\ P,\ S)$ **where**

$$P \ : \quad S \to \varepsilon \mid aSb \mid bSa \mid SS$$

(2) $L = \{w \mid w \in \{a,\ b\}^* \text{ and } |w|_a = |w|_b\}$,

prove that $\underline{L(G) = L}$.

Proof:

(I) First we prove that $\underline{L(G) \subseteq L}$.

Claim 1. For all $w \in \Sigma^*$ **such that** $S \Rightarrow^+ w, \quad |w|_a = |w|_b$.

Proof of Claim 1: (prove by induction on n**, the number of derivation steps)**

Basis : $n = 1$, $S \Rightarrow \varepsilon,\ |\varepsilon|_a = |\varepsilon|_b = 0$.

I.H. : Assume it holds for any $1 \leq k < n$, **i.e., that** $S \Rightarrow^k w,\ k < n$, **implies** $|w|_a = |w|_b$.

I.S. : Consider $S \Rightarrow^n w,\ n > 1$. **There are three cases concerning the first step of the derivation.**

Case 1 : $\underline{S \Rightarrow aSb} \Rightarrow^{n-1} aw'b.$

So, $S \Rightarrow^{n-1} w'$.

By I.H. $|w'|_a = |w'|_b$. **This implies** $|w|_a = |w|_b$.

Case 2 : $\underline{S \Rightarrow bSa} \Rightarrow^{n-1} bw'a = w$

$S \Rightarrow^{n-1} w'$ **implies** $|w'|_a = |w'|_b$ **by I.H.**

So, $|w|_a = |w|_b$.

Case 3 : $\underline{S \Rightarrow SS} \Rightarrow^{n-1} w_1w_2 = w,$ **where** $S \Rightarrow^i w_1,\ S \Rightarrow^j w_2$ **for** $i,\ j < n$.

By I.H., $|w_1|_a = |w_1|_b,\ |w_2|_a = |w_2|_b$.

therefore, $|w|_a = |w|_b$.

So, $w \in L(G)$ **implies** $w \in L$**.**

(II) Prove that $\underline{L \subseteq L(G)}$**.**

<u>Claim 2</u> if $w \in L$**, then** $S \Rightarrow^+ w$ **in** G**.**

<u>Proof of Claim 2</u>: By induction on the number of a**'s in** w**.**

Basis : $|w|_a = 0$**.** **Then** $w = \varepsilon$**.** $S \Rightarrow \varepsilon$**.**

I.H. : Assume that if $|w|_a < n$ **and** $w \in L$
then $S \Rightarrow^+ w$**.**

I.S. : $|w|_a = n$**.**

Case 1 : $w = axb$. **Thus** $|x|_a < n$.
Hence $S \Rightarrow^+ x$ **by I.H. . So,**
$S \Rightarrow aSb \Rightarrow^+ axb = w$ **in** G.

Case 2 : $w = bxa$. **As above.**

Case 3 : $w = axa$. **Then**
$\underline{w = yz \text{ for some } y,\ z \text{ s.t.}}$
$\underline{|y|_a = |y|_b,\ |z|_a = |z|_b.}\ y,\ z \neq \varepsilon$
(We prove it later).
Hence, $S \Rightarrow^+ y$ **and** $S \Rightarrow^+ z$ **by I.H.**
Therefore, $S \Rightarrow SS \Rightarrow^+ yz = w$.

Case 4 : $w = bxb$. **As above.**

So, $L \subseteq L(G)$**. We now conclude that** $L = L(G)$**.**

Claim **If** $w \in L$ $(i.e.|w|_a = |w|_b)$ **and** $w = axa$, **then** $w = yz$ **such that** $|y|_a = |y|_b$ **and** $|z|_a = |z|_b$**.**

Proof:

Let $w = c_1c_2c_3 \ldots c_{2n}$**,** $c_i \in \{a, b\}$**, for all** $1 \leq i \leq 2n$**.**

Let $w_i = c_1c_2 \ldots c_i$ **for** $1 \leq i \leq 2n$**.**

Now, consider the sequence:

$|w_1|_a - |w_1|_b$**,** $|w_2|_a - |w_2|_b$**, . . . ,**$|w_{2n-1}|_a - |w_{2n-1}|_b$

Obviously, $|w_1|_a - |w_1|_b = 1$ **and**

$$|w_{2n-1}|_a - |w_{2n-1}|_b = -1.$$

Let k, be the smallest integer s.t. $|w_k|_a - |w_k|_b = -1$. **Then** $2 < k \leq 2n - 1$**.**

Clearly, $|w_{k-1}|_a - |w_{k-1}|_b = 0$

Let $y = w_{k-1}$**,** $z = c_kc_{k+1} \ldots c_{2n}$.

Therefore, $w = yz$ **and** $|y|_a = |y|_b$**,** $|z|_a = |z|_b$**.**

Regular grammars

Definition A CFG, $G = (N, \Sigma, P, S)$ is said to be linear if every production in P is either of the forms

$$A \to x,\ x \in \Sigma^*,\ \ A \in N,$$

or $$A \to xBy,\ x,\ \ y \in \Sigma^*,\ \ B \in N$$

Definition A CFG $G = (N, \Sigma, P, S)$ is said to be right linear if every production in P is of one of the forms:

$$A \to x,\ x \in \Sigma^*, A \in N,$$

$$A \to xB,\ B \in N.$$

Definition . . . Left linear . . .

$$A \to x,\ \ldots$$

$$A \to Bx,\ \ldots$$

Definition A CFG G is said to be regular if it is right linear or left linear

Since left linear grammars define the same set of languages as right linear grammars, by regular grammars we usually mean right linear grammars.

The following definition is equivalent to the above one.

Definition A CFG $G = (N, \Sigma, P, S)$ is regular if every production in P is of one of the following forms:

$$\begin{array}{ll} A \to a, & a \in \Sigma \cup \{\varepsilon\},\ A \in N \\ A \to aB, & B \in N. \end{array}$$

Example

1. $S_1 \to \varepsilon \mid aS_1 \mid bB$

$B \to \varepsilon \mid bB$

2. $S_2 \to AA$

$A \to aA \mid \varepsilon$

<u>Lemma 1</u> For every regular grammar $G = (N, \Sigma, P, S)$, there is an ε-NFA M such that $L(G) = L(M)$.

<u>Proof</u>: Let $M = (Q, \Sigma, \delta, s, \{f\})$ where

$$Q = N \cup \{f\}$$

$$s = S$$

$$\delta = \{(A, a, B) : A \to aB \textbf{ is in } P\}$$

$$\cup\{(A, a, f) : A \to a \textbf{ is in } P\}$$

<u>Claim</u> $A \Rightarrow^* w$ in G iff $Aw \vdash^+ f$ in M.

<u>Proof</u> by induction on derivation lenght.

<u>Example</u> $G = (N, \Sigma, P, S)$ when P:
$S \to aS \mid \varepsilon \mid bB$**,** $\qquad B \to bB \mid \varepsilon$**,**

Construct an ε-NFA M such that
$L(M) = L(G)$.

<u>Lemma 2</u> For every NFA $M = (Q, \Sigma, \delta, s, F)$ there is a regular grammar G such that $L(G) = L(M)$.

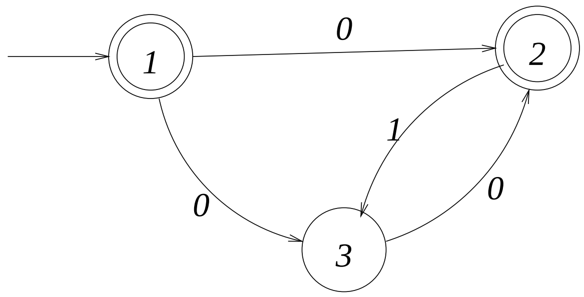
<u>Proof</u>: Let $G = (N, \Sigma, P, S)$ where

$$N = Q$$

$$S = s$$

$$P = \{p \rightarrow aq : (p, a, q) \in \delta\}$$
$$\cup \{p \rightarrow \varepsilon : p \in F\}$$

<u>Example</u> M:



Construct a regular grammar G using the above method

$G = (N, \Sigma, P, S)$
$N = \{(1), (2), (3)\}, \quad S = (1)$
$P : (1) \rightarrow 0(2), \quad (1) \rightarrow 0(3), \quad (2) \rightarrow 1(3)$
$(3) \rightarrow 0(2), \quad (1) \rightarrow \varepsilon, \quad (2) \rightarrow \varepsilon$

Claim For any $w \in \Sigma^*$, $pw \vdash^* q$ in M iff $p \Rightarrow^* wq$ in G.

The claim implies $sw \vdash^* f$ in M, for some $f \in F$, iff $s \Rightarrow^* wf \Rightarrow w$ in G.
Therefore, $w \in L(M)$ iff $w \in L(G)$. $\square$

Theorem 1 Regular grammars define exactly the family of regular languages (i.e. DFA languages).
Proof: By Lemma 1 and 2.

Theorem 2 $\mathcal{L}_{REG} \subset \mathcal{L}_{CF}$
Proof : By Theorem 1 and the fact that

$$\{a^n b^n \mid n \geq 0\} \text{ is not regular.}$$

Derivations of CFG's

Definition Let $G = (N,\ \Sigma,\ P,\ S)$ be a CFG. A word $\alpha \in V^*$ (i.e., $\alpha \in (N \cup \Sigma)^*$) is a sentential form if $S \Rightarrow^* \alpha$.
If $\alpha \in \Sigma^*$, then α is also called a sentence.

Now we consider the derivation of a CFG $G_1 = (N_1,\ \Sigma_1,\ P_1,\ S_1)$ where P_1:

$$S_1 \rightarrow T \mid S_1 + T$$

$$T \rightarrow F \mid F * T$$

$$F \rightarrow a \mid (S_1)$$

To derive $a + a$, we have

$$S_1 \Rightarrow S_1 + T \Rightarrow T + T$$

and from $T + T$ there are 6 ways to derive $a + a$:

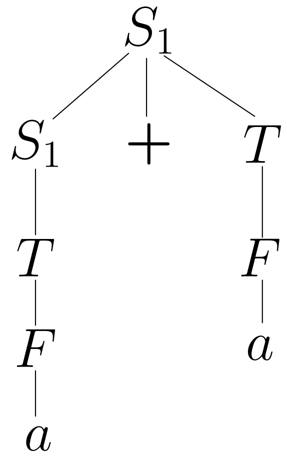
(1) $T + T \Rightarrow F + T \Rightarrow a + T \Rightarrow a + F \Rightarrow a + a$
(2) $T + T \Rightarrow F + T \Rightarrow F + F \Rightarrow a + F \Rightarrow a + a$
(3) $T + T \Rightarrow F + T \Rightarrow F + F \Rightarrow F + a \Rightarrow a + a$
(4) $T + T \Rightarrow T + F \Rightarrow T + a \Rightarrow F + a \Rightarrow a + a$

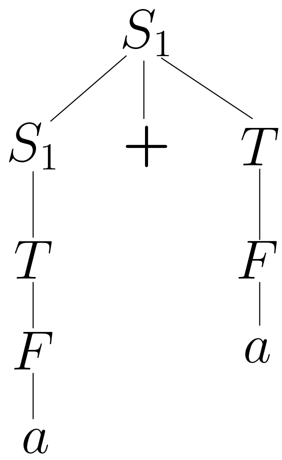
(5) $T + T \Rightarrow T + F \Rightarrow F + F \Rightarrow F + a \Rightarrow a + a$

(6) $T + T \Rightarrow T + F \Rightarrow F + F \Rightarrow a + F \Rightarrow a + a$

The above derivations can be represented by rooted directed trees.

For the 6 sequences of derivations, we have:





Consider another CFG $G_2 = (N_2, \Sigma_2, P_2, S_2)$ **where**

$P_2 \ : \ S_2 \rightarrow S_2 + S_2 \mid a$

To derive $a + a + a$**, we have**

$\underline{S_2} \Rightarrow \underline{S_2}+S_2 \Rightarrow \underline{S_2}+S_2+S_2 \Rightarrow a+S_2+S_2 \Rightarrow^2 a+a+a$

$\underline{S_2} \Rightarrow S_2+\underline{S_2} \Rightarrow \underline{S_2}+S_2+S_2 \Rightarrow a+S_2+S_2 \Rightarrow^2 a+a+a$

Hence, different derivation sequences do not necessarily represent different structures.

To solve this problem we use

(1) syntax trees, or

(2) canonical derivations.

Syntax trees

Definition **Let** $G = (N,\ \Sigma,\ P,\ S)$ **be a CFG. Then** T **is a syntax tree with respect to** G **if every node** u **in** T **satisfies the following conditions:**

(1) If u **is an external node, then it is labelled with a symbol in** $N \cup \Sigma$ **or** ε**, and in the latter case it is the only child of its parent.**

(2) Otherwise, u **is labelled with a symbol** A **in** N**, and it has** k **children,** $k \geq 1$**, labelled with** $X_1 \ldots X_k$**, from left to right and,**

$$A \rightarrow X_1 \ldots X_k \textbf{ is in } P.$$

(2) Canonical derivations

A rewriting (derivation) step is called a rightmost rewriting (derivation) step if the rightmost nonterminal is being rewritten.

For example, $aS\underline{S}b \Rightarrow aS\underline{bSa}b$ **is a rightmost derivation step.**

A sequence of rightmost derivation steps is called a rightmost derivation (sequence).

Leftmost derivations are similarly defined.

Notations

Rightmost derivations:

$$\Rightarrow_R \qquad \Rightarrow_R^+ \qquad \Rightarrow_R^* \qquad \Rightarrow_R^n$$

Leftmost derivations:

$$\Rightarrow_L \qquad \Rightarrow_L^+ \qquad \Rightarrow_L^* \qquad \Rightarrow_L^n$$

Let $G = (N,\ \Sigma,\ P,\ S)$ be a CFG.

Let $\underline{w} \in L(G)$.

- **If there are two distinct derivation trees (leftmost or rightmost derivations) that derive w by G, then w is said to be <u>ambiguous</u> with respect to G.**

- **G is said to be <u>ambiguous</u> if there is at least one word in $L(G)$ that is ambiguous.**

- **A <u>context-free language</u> is ambiguous if for all CFGs G, with $L(G) = L$, G is ambiguous.**

Simplifications and Normal Forms

Redundant Symbols

Definition **A symbol $X \in V$ ($V = N \cup \Sigma$) is said to be a terminating symbol if**

(1) X **is a terminal; (i.e,** $X \in \Sigma$**)**

(2) $X \to \alpha \ \in P$ **and** α **consists solely of terminating symbols.**

Example **Let** G **be described by**

$$S \to ASB \mid BSA \mid SS \mid aS \mid \varepsilon$$
$$A \to AB \mid B$$
$$B \to BA \mid A$$

(a) Find the set of terminating symbols of G

(1) $\{a\}$**,** **(2)** $\{a, S\}$

(b) Eliminate non-terminating symbols

$$S \to SS \mid aS \mid \varepsilon$$

<u>Definition</u> Let $G = (N, \Sigma, P, S)$ be a CFG. A symbol $X \in V$ is said to be <u>reachable</u> if $S \Rightarrow^* \alpha X \beta$, for $\alpha, \beta \in V^*$.

<u>Example</u> Let G be

$$S \to aS \mid SB \mid SS \mid \varepsilon$$
$$A \to ASA \mid C$$
$$B \to b$$

(a) Mark all the reachable symbols

$\{S\}$

$\{S,\ a,\ B\}$

$\{S,\ a,\ B,\ b\}$

(b) Eliminate unreachable symbols

$$S \to aS \mid SB \mid SS \mid \varepsilon$$
$$B \to b$$

Summary of "Redundant Symbols"

Redundant symbols:

(1) nonterminating symbols

nonterminal symbols that do not derive any terminal word.

(2) unreachable symbols

symbols not appear in any sentential form

Definition A CFG G is said to be reduced if G does not contain redundant symbols.

Theorem Given a CFG $G = (N, \Sigma, P, S)$, an equivalent reduced CFG $G' = (N', \Sigma', P', S)$ can be constructed such that $N' \subseteq N$, $\Sigma' \subseteq \Sigma$ and $P' \subseteq P$.

Empty Productions

Definition **An empty-production is a production of the form**

$$\underline{A \to \varepsilon} \ .$$

Empty productions are also called ε**-productions, null-productions.**

△ **A nonterminal** B **is called a** ε**-nonterminal if** $B \Rightarrow^{+} \varepsilon$**.**

△ **Use a marking alghorithm to find all the** ε**-nonterminals.**

Example **Let** $G = (N, \Sigma, P, S)$ **be**

$$\begin{aligned} &S \to aSaS \mid SS \mid bA \\ &A \to BC \\ &B \to \varepsilon \\ &C \to BB \mid bb \mid aC \mid aCbA \end{aligned}$$

Find all the ε**-nonterminals in** G**.**

(1) $\{B\}$**,** **(2)** $\{B,\ C\}$**,** **(3)** $\{B,\ C,\ A\}$

Algorithm to remove ε-productions

i) Use the marking alghorithm to find all the ε-nonterminals.

ii) For every ε-nonterminal A in N do

for every $B \to \beta$ in P with $|\beta|_A \neq 0$ do

Let $\beta = \beta_0 A \beta_1 \ldots \beta_{t-1} A \beta_t$, where $\beta_0, \beta_1, \ldots, \beta_t$ do not contain A.

Replace $B \to \beta$ in P with all the productions

$$\{B \to \beta_0 X_1 \beta_1 \ldots \beta_{t-1} X_t \beta_t \mid X_i \in \{\varepsilon, A\}\}$$

iii) Reduce the new grammar (i.e remove all the redundant symbols).

Theorem **Let G be a reduced CFG $G = (N, \Sigma, P, S)$. Then there exists a ε-equivalent CFG $G' = (N', \Sigma, P', S)$ that is also reduced and ε-free.**

ε-equivalent: $L(G) - \{\varepsilon\} = L(G') - \{\varepsilon\}$

ε-free: no ε-productions.

Chomsky Normal Form

Definition **A CFG G is said to be in Chomsky normal form if it only has productions of the forms:**

$$i)\ A \to a, \qquad a \in \Sigma;$$
$$ii)\ A \to BC, \qquad B,\ C \in N.$$

An arbitrary CFG G may have productions of the following forms:

(Assume: G doesn't have ε-productions, G is reduced)

$i)\ A \to a;$

$ii)\ A \to BC;$

$iii)\ A \to B;$ **(unit-productions)**

$iv)\ A \to \alpha,$ $|\alpha| > 2$ **and** $\alpha \in V^+;$

$v)\ A \to \alpha,$ $|\alpha| = 2$ **and** $\alpha \notin N^2;$

We are going to show that $iii)$, $iv)$ and $v)$ can be changed to $i)$ and $ii)$.

Unit-production removal

While there is a unit-production $A \to C$ in P **do**
if $A = C$ **then** remove $A \to C$ from P
else replace $A \to C$ with all productions
$A \to \alpha : C \to \alpha$ in P.

Does this alghorithm always terminate? Why?

$\triangle$ **Reduced, ε-free CFG**
$\Longleftarrow transfer \Longrightarrow$ **reduced, ε-free, unit-free CFG**

Long production removal

Let $Maxrhs(G) = max\{|\alpha| : A \to \alpha \text{ in } P\}$.
Claim Let $k = Maxrhs(G)$ and $k \geq 3$. Then we can construct an equivalent G' such that $Maxrhs(G') < k$.

Proof (outline)

If $A \to \alpha$ **is in** P **and** $|\alpha| = k \geq 3$**,**
then replace it with

$i)$ $A \to \alpha_1[A\alpha]$
$ii)$ $[A\alpha] \to \alpha_2$

where $[A\alpha]$ **is a new nonterminal,** $\alpha = \alpha_1\alpha_2$
and $|\alpha_1| = \lfloor |\alpha|/2 \rfloor$**,** $\quad |\alpha_2| = \lceil |\alpha|/2 \rceil$**.**
Note that if $i)$ **is used in a derivation**
then $ii)$ **must be used.**
So, G' **is equivalent to** G**.**

Note: If G **is** ε**-free, unit-free then**
G' **is a** ε**-free and unit-free.**

$\triangle$ **By iterating the above approach, we can get a CFG** G **s.t.** $Maxrhs(G) = 2$**.**

Changing to CNF

Now, we have all the productions in the forms i), ii) and v) where:

$$v):\begin{cases} A \to aB \\ A \to Ba \\ A \to ab \end{cases}$$

$$\begin{array}{llll} A \to aB & \Rightarrow & A \to \overline{a}B, & \overline{a} \to a \\ A \to Ba & \Rightarrow & A \to B\overline{a}, & \overline{a} \to a \\ A \to ab & \Rightarrow & A \to \overline{a}\overline{b}, & \overline{a} \to a,\ \overline{b} \to b; \end{array}$$

where $\overline{a},\ \overline{b}$ are new nonterminals.

The CFG's in CNF can generate all the CFL's.

Summary

Given an arbitrary CFG $G = (N, \Sigma, P, S)$**,**
we can construct a ε**-equivalent CFG** G'
s.t. G' **is in CNF by the following steps:**

1) reduction;

i) remove nonterminating symbols,
ii) remove unreachable symbols;

2) remove ε**-productions; (may reduce again)**

3) remove unit-productions;

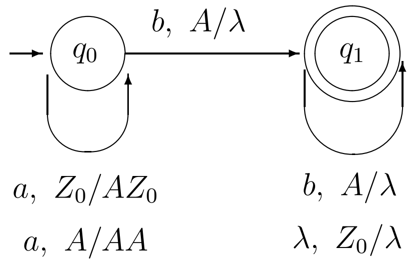
4) remove long productions; (≥ 3)

5) change to CNF

2. Pushdown Automata

Definition **A PDA A is a 7-tuple $(Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ where**

- Q **: a finite set of states;**
- Σ **: input alphabet;**
- Γ **: stack alphabet;**
- δ **:** $Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \times Q \times \Gamma^*$ **transition relation;**
- $q_0 \in Q$ **: the initial state;**
- $Z_0 \in \Gamma$ **: the bottom-of-stack symbol;**
- $F \subseteq Q$ **: set of final states;**



By Final State : $\{a^i b^j \mid i \geq j \geq 1\}$

Instantaneous descriptions (IDs)

$$(\underbrace{\quad q \quad}_{\text{current state}}, \overbrace{\quad x \quad}^{\text{remaining part of the input}}, \underbrace{\quad \alpha \quad}_{\text{current content of the stack}})$$

An ID describes a configuration of a PDA.

Example

$$\overbrace{(q_0, aab, Z_0)}^{\text{ID for the initial configuration}} \vdash (q_0, ab, AZ_0) \vdash (q_0, b, AAZ_0)$$
$$\underbrace{\vdash (q_1, \varepsilon, AZ_0)}_{\text{ID for an accepting configuration}}$$

Acceptance methods of PDA:

(1) by final state

$$T(A) = \{w \mid (q_0, w, Z_0) \vdash^* (q_f, \varepsilon, \alpha),\ q_f \in F\}$$

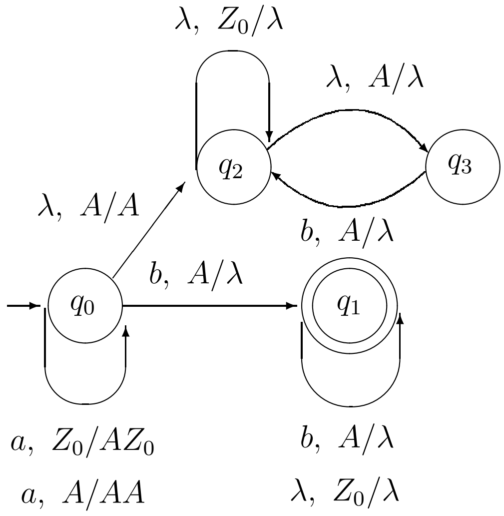
(2) by empty stack

$$N(A) = \{w \mid (q_0, w, Z_0) \vdash^* (q, \varepsilon, \varepsilon)\}$$

(3) by both final state and empty stack

$$L(A) = \{w \mid (q_0, w, Z_0,) \vdash^* (q_f, \varepsilon, \varepsilon),\ q_f \in F\}$$

Example



$$N(B) = \{a^i b^i \mid i > 0\} \cup \{a^{2i} b^i \mid i > 0\}$$

$$L(B) = \{a^i b^i \mid i > 0\}$$

$$T(B) = \{a^i b^j \mid i \geq j > 0\}$$

Deterministic Context-free Languages

Definition **A PDA** $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ **is deterministic if**

1) **for each** q **in** Q, a **in** $\Sigma \cup \{\varepsilon\}$, **and** $X \in \Gamma$, $\delta(q, a, X)$ **contains at most one element,**

2) **whenever** $\delta(q, a, X)$ **is nonempty for some** $a \in \Sigma$, **then** $\delta(q, \varepsilon, X)$ **is empty.**

Note that

— **DPDA allow** $\varepsilon - transitions$.

— **Each transition is determined by the current state, the input symbol, and the top-of-stack symbol.**

So, for each pair of a state and an input symbol, there can be several transitions, one for each stack symbol.

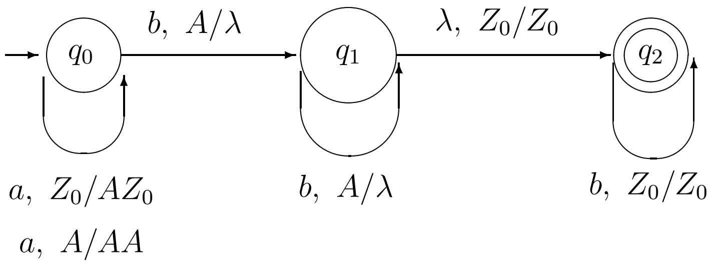
— $\delta(q, \varepsilon, X)$ **should not be defined if** $\delta(q, a, X)$ **is defined for any** $a \in \Sigma$.

Let $\mathcal{T}_{DPDA}$, $\mathcal{N}_{DPDA}$, and $\mathcal{L}_{DPDA}$ **denote the sets of languages accepted by DPDA with acceptance by "final state", "empty stack", and "final state and empty stack", respectively.**

Then $\mathcal{N}_{DPDA} = \mathcal{L}_{DPDA} \subset \mathcal{T}_{DPDA}$

Example

$L = \{a^m b^n | m \leq n,$ **and** $m, n > 0\}$
Then $L = T(A)$ **where** A**:**



But $L \notin \mathcal{N}_{DPDA}$

Definition The family of deterministic context-free languages is the set of all languages accepted by DPDA with acceptance by final state.

The family of DCFLs is a proper subset of the family of CFLs.

Examples

The following CFLs are not DCFLs

1. $\{a^n b^n \mid n \geq 0\} \cup \{a^n b^{2n} \mid n \geq 0\}$

2. $\{ww^R \mid w \in Z^*\}$

3. $\overline{\{ww \mid w \in Z^*\}}$

The family of DCFLs is closed under

(1) complementation,
(2) intersection with regular sets,

not closed under

(1) union
(2) intersection.

3. CFL Pumping Lemma & Closure Properties

△ Pumping Lemma

Let $L = L(G)$ **and** $G = (N, \Sigma, P, S)$ **be a** ε**-free, unit-free CFG, such that**
$m = max(\{|\alpha| \mid A \rightarrow \alpha \in P\})$ **and** $p = 1 + m^{\#N+1}$.

Then, for all words z **in** $L(G)$ **such that** $|z| \geq p$, z **has a derivation sequence**

$S \Rightarrow^{*} uAv \Rightarrow^{+} uxAyv \Rightarrow^{+} uxwyv = z$

for some A **in** N **and some** u, v, w, x, y **in** Σ^{*} **such that**

i) $|xwy| < p$;

ii) $|xy| \geq 1$;

iii) $ux^{i}wy^{i}v$ **is in** L, **for all integers** $i \geq 0$.

Example (Use of CFG P.L.)

Prove that $L = \{a^i b^i c^i \mid i \geq 1\}$ is not a CFL.

Proof

Assume L is a CFL.

Then there exists a ε-free unit-free CFG G s.t. $L = L(G)$. Let p be the constant for G defined in P.L By P.L, all words z with $|z| \geq p$ can be decomposed into $z = uxwyv$ s.t

- **i)** $|xwy| < p$
- **ii)** $|xy| \geq 1$
- **iii)** $ux^i wy^i v \in L$, **for all** $i \geq 0$.

Therefore, to obtain a contradiction, it is sufficient to give one word that for all decompositions, conditions i), ii), and iii) cannot be satisfied at the same time.

Consider $z = a^p b^p c^p$. Obviously, $|z| > p$. Since $|xwy| < p$, xwy is in $a^+ \cup b^+ \cup c^+ \cup a^+ b^+ \cup b^+ c^+$. (The only possibilities)

<u>Case 1</u> <u>xwy is in a^+</u>

Then $xy = a^k$, for all $1 \leq k < p$.
Consider $ux^0wy^0v = uwv = a^{p-k}b^pc^p$.
Since $k \geq 1$, there are less a's than b's and c's. $uwv \notin L$

<u>xwy in b^+</u> or <u>in c^+</u> are similar.

<u>Case 2</u> <u>xwy is in a^+b^+</u>

(1) x is in a^+b^+ (or y is in a^+b^+)
Then ux^2wy^2v is not in L since it has a's following b's.

(2) x in a^*, y in b^*. Since $|xy| \geq 1$, x, y cannot all be ε. Then $ux^0wy^0v \notin L$ since there are less a's or b's than c's.

The case of <u>xwy in b^+c^+</u> is similar.

Since all the decompositions fail to satisfy all the conditions i), ii) and iii), $z = a^pb^pc^p$ contradicts P.L. Therefore, L is not in CFL.

Proof of CFG P.L

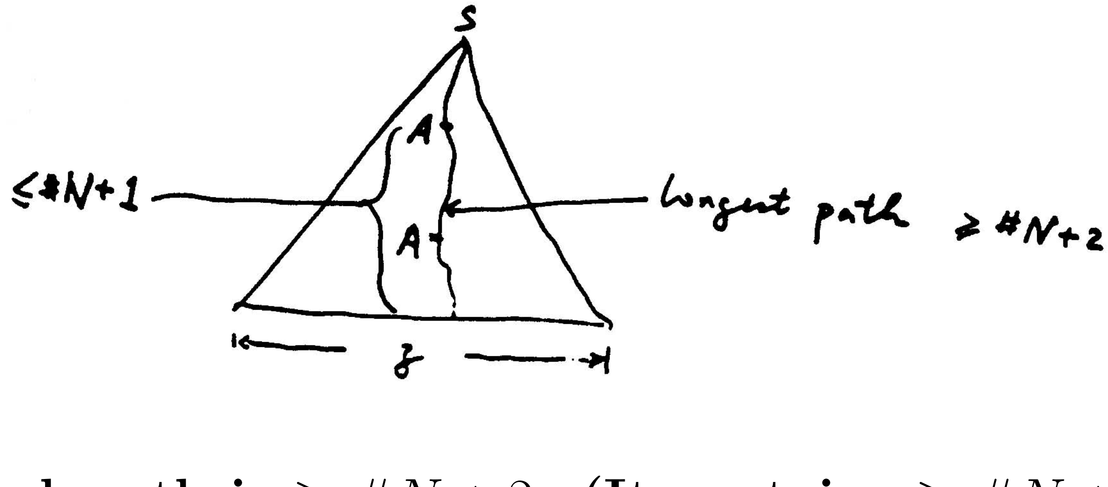
It is easy to prove that an m-ary tree with $\geq m^h$ external nodes has height $>\ h$.

Therefore,

i) if a syntax tree for G has a yield $> m^h$ ($m = maxrhs(G)$), then its height is $>\ h$.

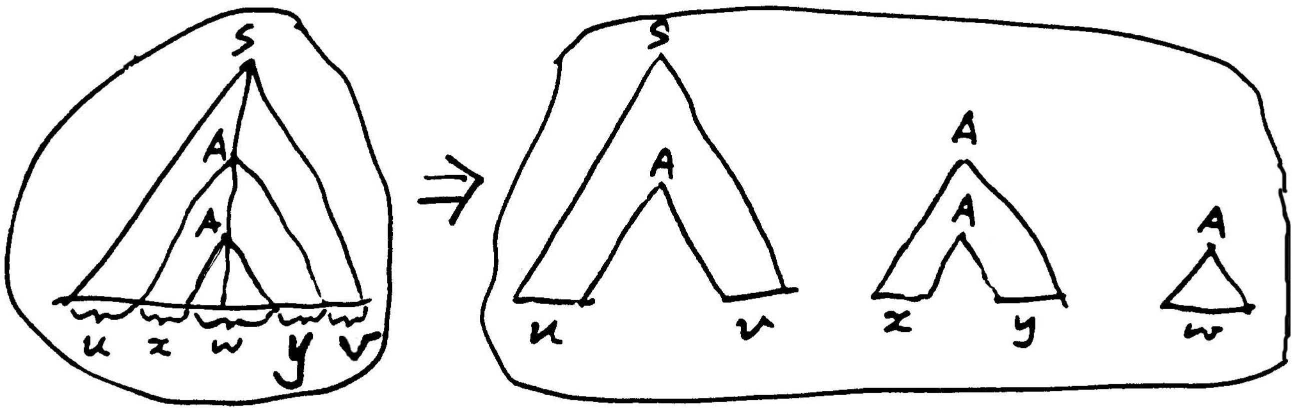
ii) if it has height $\leq h$, then its yield has a length $\leq m^h$.

Consider a word z with $|z| \geq p > m^{\#N+1}$. Then any syntax tree T for z satisfies $ht(T) > \#N+1$. Consider a longest path from the root to frontier.

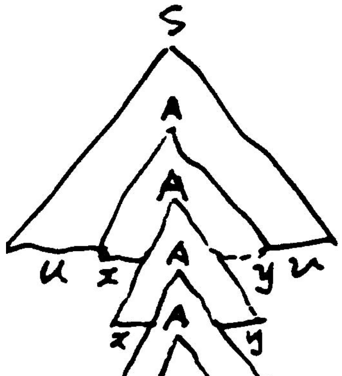


Its length is $\geq \#N+2$. (It contains $\geq \#N+3$ symbols.)

Consider the lowest $\#N + 1$ nonterminal symbols. By Pigeonhole principle, there must be some nonterminal $\underline{A}$ that appears at least twice among these $\#N + 1$ nonterminals on this path. This provides a decomposition of z.



$$S \Rightarrow^* uAv \Rightarrow^+ uxAyv \Rightarrow^+ uxwyv$$



$$S \Rightarrow^* uAv \Rightarrow^+ uxAyv \Rightarrow^+ ux^iAy^iv \Rightarrow^+ ux^iwy^iv$$

Now, consider the three conditions.

i) **Since the upper** $\underline{A}$ **is at a distance of at most** $\#N + 1$ **from the frontier,** $\underline{|xwy| \leq m^{\#N+1} \leq p}$.

ii) **Since** G **is** ε**-free, unit-free,** $|xy| \geq 1$**.**

iii) **As discussed on the last page.**

$\triangle$ Closure Properties

We show that $\mathcal{L}_{CF}$ is closed under $\cup$, $\bullet$, $*$, but not under $\cap$ and $^{-}$.

1. Union

$L_1, L_2 \in \mathcal{L}_{CF}$ **(i.e L_1, L_2 are CFLs).**
Show that $L = L_1 \cup L_2$ is CF.

Proof:

$G_1 = (N_1, \Sigma_1, P_1, S_1)$, $G_2 = (N_2, \Sigma_2, P_2, S_2)$
Assume $N_1 \cap N_2 = \emptyset$. Construct

$G =$
$(N_1 \cup N_2 \cup \{S\}), \Sigma_1 \cup \Sigma_2, P_1 \cup P_2 \cup \{S \to S_1 | S_2\}, S)$.

Then $L(G) = L(G_1) \cup L(G_2)$

2. Catenation

$$L_1, L_2 \in \mathcal{L}_{CF} \Rightarrow L_1 L_2 \in \mathcal{L}_{CF}$$

$G_1 = (N_1, \Sigma_1, P_1, S_1)$ **,** $G_2 = (N_2, \Sigma_2, P_2, S_2)$

$$G = (N_1 \cup N_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, \\ P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}, S))$$

$$L(G) = L(G_1) \bullet L(G_2)$$

3. *

$L_1 \in \mathcal{L}_{CF} \Rightarrow L_1^* \in \mathcal{L}_{CF}$ $\qquad (L(G_1))^*$

$$S \rightarrow S_1 S | \varepsilon$$

4. Intersection

$$L_1, L_2 \in \mathcal{L}_{CF} \not\Rightarrow L_1 \cap L_2 \in \mathcal{L}_{CF}$$

$L = \{a^i b^i c^i \mid i \geq 0\}$ **is not in CF**

$$L_1 = \{a^i b^j c^k \mid i = j, i, j, k \geq 0\}$$

$$L_2 = \{a^i b^j c^k \mid j = k, i, j, k \geq 0\}$$

$$L_1 \cap L_2 = L$$

5. **Complementation**

$L_1 \in \mathcal{L}_{CF} \not\Rightarrow \overline{L_1} \in \mathcal{L}_{CF}$

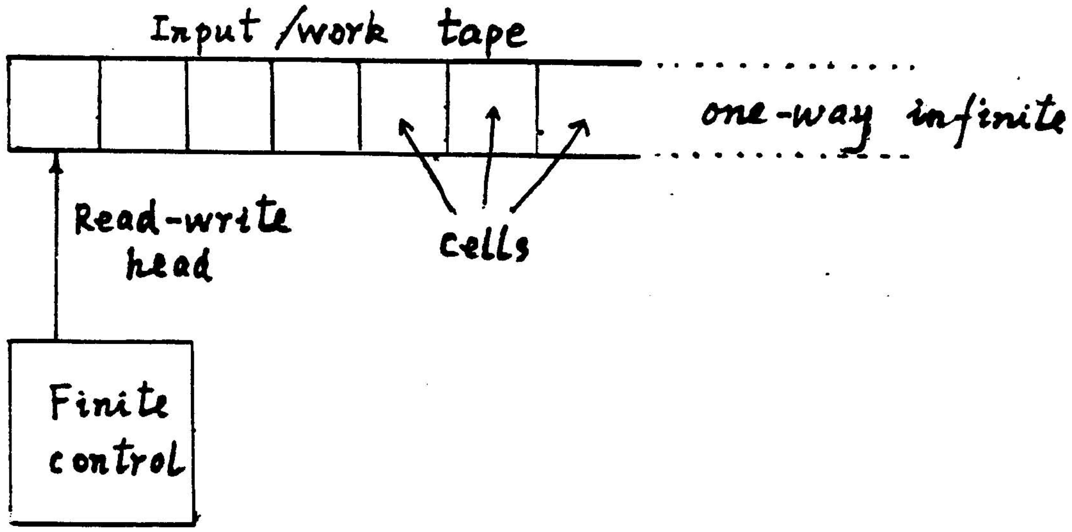
Proof:

Assume $\mathcal{L}_{CF}$ is closed under $^-$.
Consider two arbitrary CFLs L_1, L_2.
$L = L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

L **is CF**

$\mathcal{L}_{CF}$ **is closed under $\cap$.**
This is a contradition.

V. TURING MACHINES



Turing machines have more features than FA and PDA.

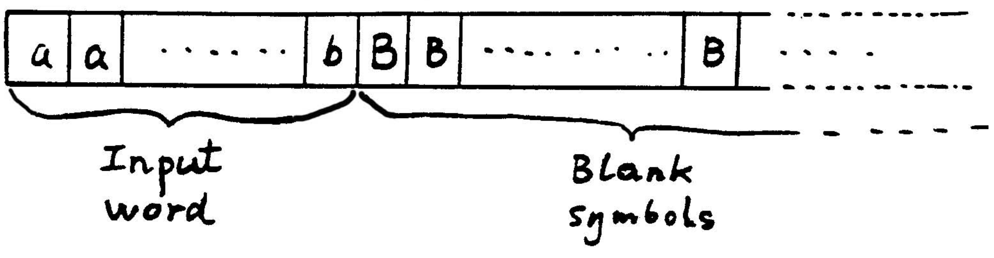
(1) The read-write head can move in either direction.

(2) It can write on the tape.

Turing machines are studied as a theoretical model of computers.

Some assumptions for TM's:

(1) At beginning, the input string (symbols) is placed at the left end of the input tape and followed by infinitely many blank symbols denoted by B's.



(2) There is only one final state denoted by "f".

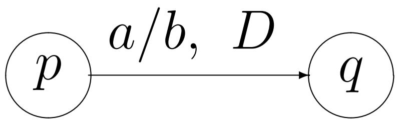
(3) A TM stops when it enters the final state "f".

<u>Definition</u> A deterministic Turing Machine (DTM) is specified by a sextuple
$(Q, \Sigma, \Gamma, \delta, s, f)$ **where**

- Q**: is a finite set of <u>states</u>;**
- Σ**: is an alphabet of <u>input symbols</u>;**
- Γ**: is an alphabet of <u>tape symbols</u>,** $\underline{\Sigma \cup \{B\} \subseteq \Gamma}$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, \varepsilon\}$ **is a <u>transition function</u>;**
- $s \in Q$ **is a <u>start state</u>;**
- $f \in Q$ **is a <u>final state</u>;**

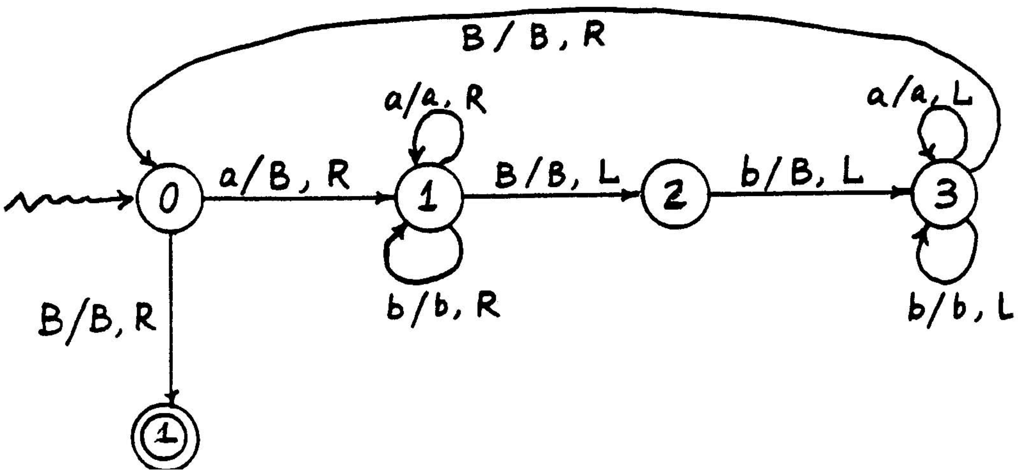
<u>State diagram</u>

$\delta(p, a) = (q, b, D)$ **is depicted graphically:**



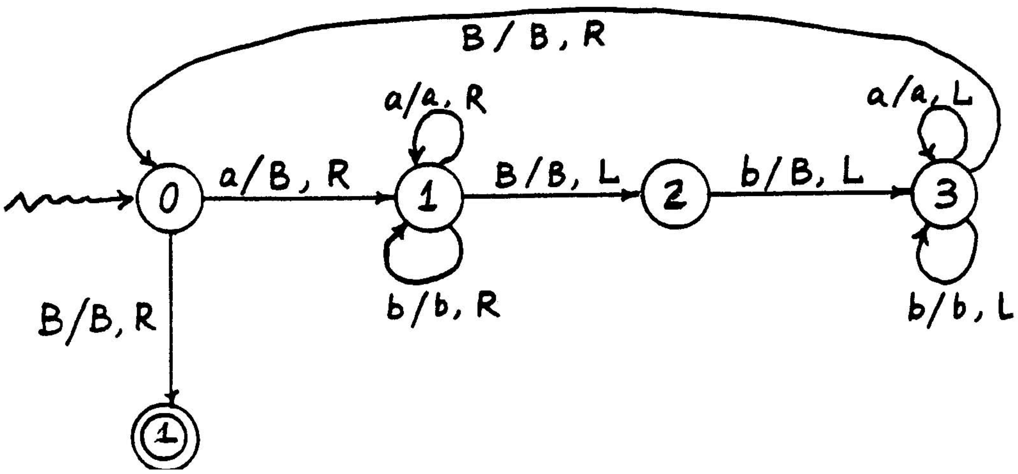
<u>Example</u> A DTM that accepts

$L = \{a^i b^i \mid i \geq 0\}$



Example **A DTM that accepts**

$L = \{a^i b^i \mid i \geq 0\}$



Configuration

A configuration is a word in

$$\Gamma^* Q \Gamma^*.$$

Strictly speaking, a configuration is a word in $\Gamma^* Q \Gamma^* (\Gamma - \{B\}) \cup \Gamma^* Q$

(Note: $Q \cap \Gamma = \emptyset$**)**

One move of a DTM

$$g_1 p h_1 \vdash g_2 q h_2 \quad \textbf{if}$$

(i) either $h_1 = A h_1'$**, for some** A **in** Γ**,** h_1' **in** Γ^* **or** $h_1 = \varepsilon$**, then** $A = B$ **and** $h_1' = \varepsilon$**;**

(ii) $\delta(p, A)$ **is defined and** $p \neq f$**;**

(iii) $\delta(p, A) = (q, A', D)$

- **(a)** $D = L$**,** $g_1 = g_1' C$ **for some** $C \in \Gamma$**, and then** $h_2 = C A' h_1'$ **(if** $g_1 = \varepsilon$**, then** M **halts)**
- **(b)** $D = R$**,** $g_1 A' = g_2$ **and** $h_2 = h_1'$
- **(c)** $D = \varepsilon$**,** $g_2 = g_1$ **and** $h_2 = A' h_1'$ **(if** $A' = B,\ h_1' = \varepsilon,$ **then** $h_2 = \varepsilon$**)**

$\vdash^i$, $\vdash^+$, $\vdash^*$ **are defined as before.**

Language acceptance

$L(M) = \{x \mid sx \vdash^* yfz,$ **for some** $y, z \in \Gamma^*\}$
$\mathcal{L}_{DTM} = \{L \mid L = L(M)$ **for some DTM** $M\}$.

A DTM can be used

(i) as a language acceptor;

(ii) to compute a function:

$$f_M : \Sigma^* \to (\Gamma - \{B\})^*$$

$$f_M(x) = y \textbf{ in } (\Gamma - \{B\})^* \textbf{ iff}$$

$$sx \vdash^* y_1 f y_2, \textbf{ where } y = y_1 y_2$$

(iii) as a decision maker.

<u>Example</u> A right shift machine

Initial state:

B**: write** B**, move –, goto** f**;**

a**: write** B**, move right, goto** A**;**

b**: write** B**, move right, goto** B**;**

A**-state:**

a**: write** a**, move right, goto** A**;**

b**: write** a**, move right, goto** B**;**

B**: write** a**, move right, goto** f**;**

B**-state:**

a**: write** b**, move right, goto** A**;**

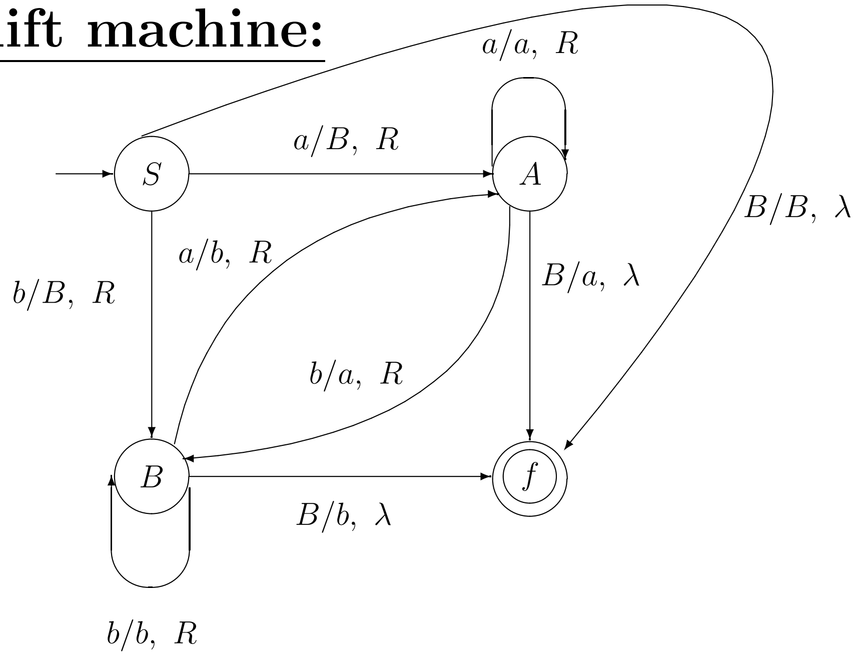
b**: write** b**, move right, goto** B**;**

B**: write** b**, move right, goto** f**;**

$|a|a|b|b|b|a|a|a|B|B|$

$|B|a|a|b|b|b|a|a|a|B|$

Right-shift machine:

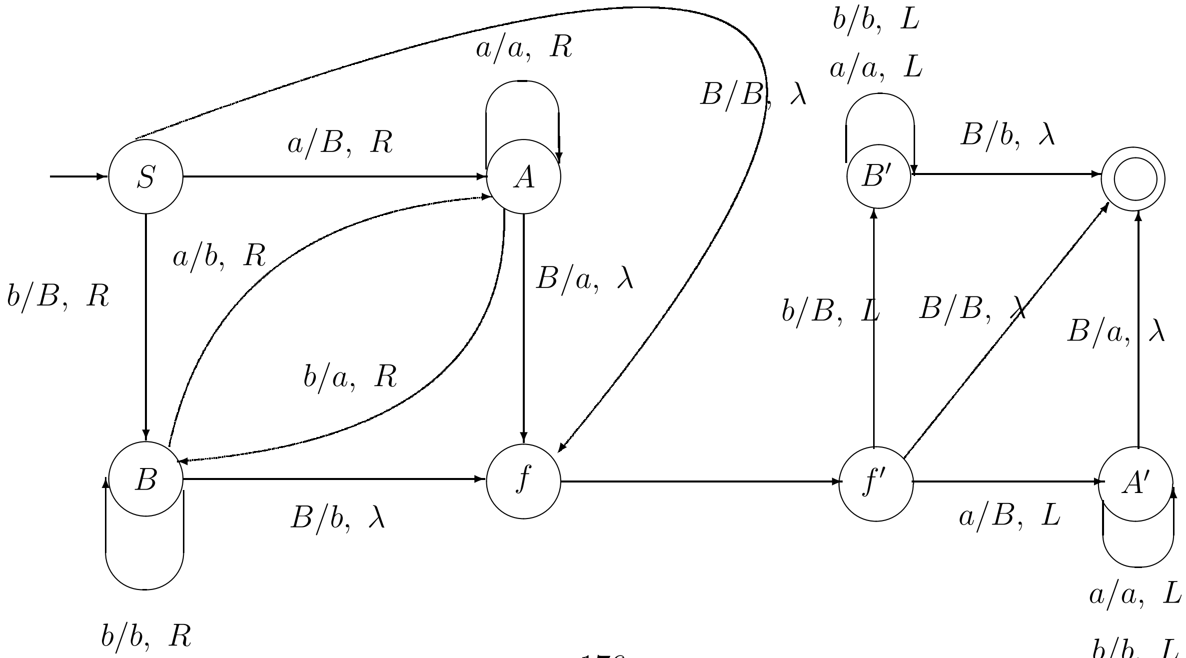


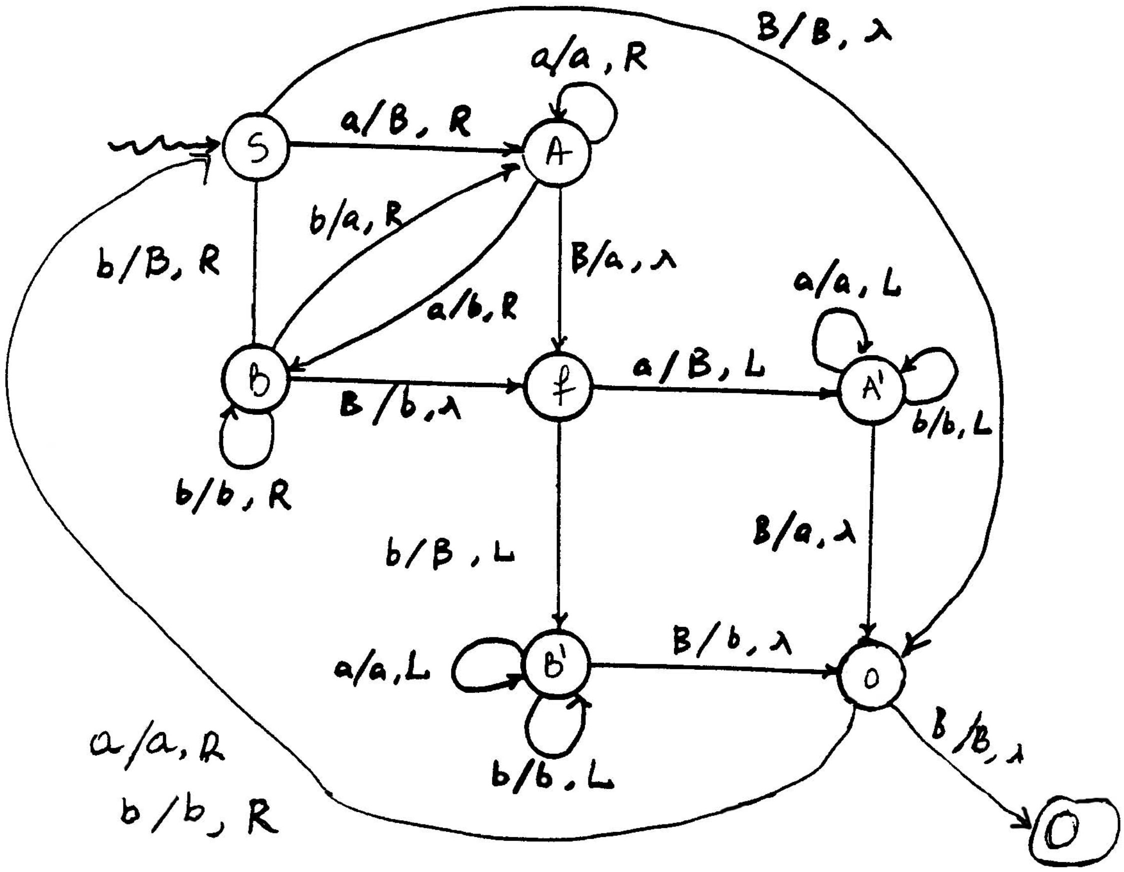
Example A cyclic right shift machine:

$$|b|a|a|b|b|b|a|B|B|\ldots\ldots$$

is transformed in:

$$|a|b|a|a|b|b|b|B|B|\ldots\ldots$$





Example A reversal machine

input: $aabba$

The Busy Beaver Problem

Consider a DTM with

— **two-way infinite tape;**

— **a tape alphabet** $\Gamma = \{1, B\}$**;**

— **n states apart from** f**.**

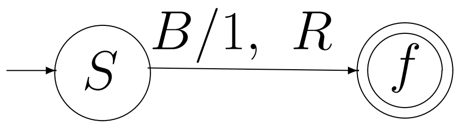
Question (by Tibor Rado)

> **How many** 1**'s can there be on entering** f**, when given the empty word as input?**

(1**'s are like twigs. Beavers build busily with twigs.)**

Define $\Sigma(n)$ **to be the maximum number of** 1**'s that can be obtained by a DTM with** n **states.**

1-state DTM:



$\Sigma(1) = 1$

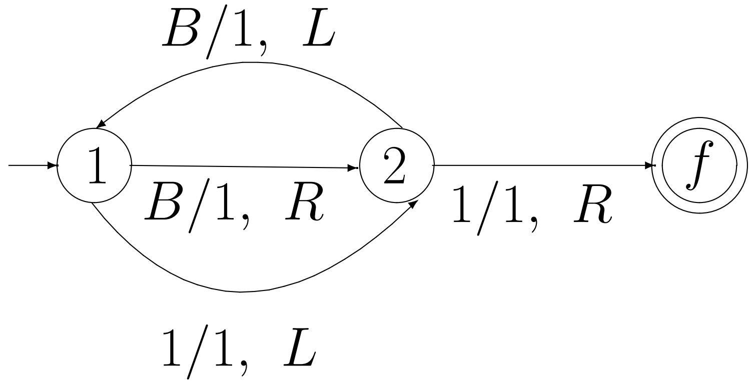
$\Sigma(2) = 4$

$\Sigma(3) = 6$

$\Sigma(4) = 13$

$\Sigma(5) = ?, \qquad \geq 1915$

2-state Machine:



$$\ldots\ldots|B|B|\underbrace{B}_{1}|B|\ldots\ldots$$

$$\ldots\ldots|B|B|1|\underbrace{B}_{2}|\ldots\ldots$$

$$\ldots\ldots|B|B|\underbrace{1}_{1}|1|\ldots\ldots$$

$$\ldots\ldots|B|\underbrace{B}_{2}|1|1|\ldots\ldots$$

$$\ldots\ldots|\underbrace{B}_{1}|1|1|1|\ldots\ldots$$

$$\ldots\ldots|1|\underbrace{1}_{2}|1|1|\ldots\ldots$$

$$\ldots\ldots|1|1|\underbrace{1}_{f}|1|\ldots\ldots$$

Definitions

Decision-making TM

A DTM $M = (Q, \Sigma, \Gamma, \delta, s, f)$ is said to be a decision making TM if y and n are in Γ and not in Σ, and for all $x \in \Sigma^*$, either $sx \vdash^* fy$ or $sx \vdash^* fn$.

Yes language of M

$$Y(M) = \{x \mid x \in \Sigma^* \text{ and } sx \vdash^* fy\}$$

No language of M

$$N(M) = \{x \mid x \in \Sigma^* \text{ and } sx \vdash^* fn\}$$

Decidability

Let $L \subseteq \Sigma^*$, $(B \notin \Sigma)$. L is decidable iff there is a decision-making TM M with $L = Y(M)$.

Recursive Languages

L is recursive iff L is decidable.

Notation

$\mathcal{L}_{REC}$ denotes the family of recursive languages.

Computability

Let $f : \Sigma^* \to \Delta^*$ be a function, where $B \notin \Sigma \cup \Delta$. f is said to be computable iff there is a DTM $M = (Q, \Sigma, \Gamma, \delta, s, f)$ with $\Delta \subseteq \Gamma$ and for all $x \in \Sigma^*$

$$\textbf{if } f(x) = y \textbf{ then}$$

$$sx \vdash^* y_1 f y_2 \quad \textbf{and} \quad y = y_1 y_2$$

for some $y_1, y_2 \in \Delta^*$.