Chapter 4. Context-Free Grammars

4.1. Introduction.

Context-free grammars (CFGs) or BNFs are very useful in describing syntax of natural or programming languages.

A CFG is essentially a collection of syntax rules which can be used to derive strings in a language.

Ex. Syntax of a class of simple arithmetic expressions involving +, ×

<expr>	⟶	<expr> + <term> |	(1)
		<term>	(2)
<term>	⟶	<term> × <factor> |	(3)
		<factor>	(4)
<factor>	⟶	(<expr>) |	(5)
		a	(6)

Starting with <expr> we can generate certain arithmetic expressions using the 6 rules above.

For example,

$$
\begin{aligned}
\langle expr\rangle &\Rightarrow \langle expr\rangle + \langle term\rangle && (1)\\
&\Rightarrow \langle expr\rangle + \langle term\rangle \times \langle factor\rangle && (3)\\
&\Rightarrow \langle expr\rangle + \langle term\rangle \times a && (6)\\
&\Rightarrow \langle expr\rangle + \langle factor\rangle \times a && (4)\\
&\Rightarrow \langle expr\rangle + a \times a && (6)\\
&\Rightarrow \langle term\rangle + a \times a && (2)\\
&\Rightarrow \langle term\rangle \times \langle factor\rangle + a \times a && (3)\\
&\Rightarrow \langle term\rangle \times a + a \times a && (6)\\
&\Rightarrow \langle factor\rangle \times a + a \times a && (4)\\
&\Rightarrow a \times a + a \times a && (6)
\end{aligned}
$$

The derivation terminates at this point and the expression obtained is $a \times a + a \times a$

Remark. $\langle expr\rangle$, $\langle term\rangle$, $\langle factor\rangle$ are nonterminals / variables, whereas (,), +, x, a are terminal symbols.

Ex. Syntax of reg. expr. over $\{0,1\}$

$$
\begin{aligned}
\langle expr\rangle &\longrightarrow (\langle expr\rangle + \langle expr\rangle) && (1)\\
\langle expr\rangle &\longrightarrow (\langle expr\rangle \langle expr\rangle) && (2)\\
\langle expr\rangle &\longrightarrow (\langle expr\rangle^{*}) && (3)\\
\langle expr\rangle &\longrightarrow 0 \mid && (4)\\
&\qquad\quad 1 \mid && (5)\\
&\qquad\quad \varepsilon \mid && (6)\\
&\qquad\quad \emptyset && (7)
\end{aligned}
$$

For example,

$$\langle expr\rangle \Rightarrow (\langle expr\rangle + \langle expr\rangle) \quad (1)$$
$$\Rightarrow ((\langle expr\rangle \langle expr\rangle) + \langle expr\rangle) \quad (2)$$
$$\Rightarrow ((0 \langle expr\rangle) + \langle expr\rangle) \quad (4)$$
$$\Rightarrow ((01) + \langle expr\rangle) \quad (5)$$
$$\Rightarrow ((01) + (\langle expr\rangle^*)) \quad (3)$$
$$\Rightarrow ((01) + (\emptyset^*)) \quad (7)$$

Ex: How to generate $\{0^k \# 1^k \mid k \geq 0\}$?

$$S \longrightarrow 0S1 \mid \# \qquad (1) \quad (2)$$

For example,

$$S \Rightarrow 0S1 \Rightarrow 00S11 \ (= 0^2S1^2)$$
$$\Rightarrow \cdots \Rightarrow 0^kS1^k$$
$$\Rightarrow 0^k \# 1^k.$$

Note that the grammar

$$S \longrightarrow A \# B$$
$$A \longrightarrow 0A \mid \varepsilon$$
$$B \longrightarrow 1B \mid \varepsilon$$

generates the language

$$\{0^m \# 1^n \mid m, n \geq 0\}$$

In defining a CFG we need to specify

(1) The set of terminal symbols

(2) The set of nonterminals/variables
(Note (1) and (2) must be disjoint)

(3) The set of rules/productions

(4) The start symbol that begins derivations.

4.2. Context-Free Grammars

Def. A **context-free grammar** (CFG) is a 4-tuple $G = (V, \Sigma, R, S)$ where:

(1) V is a finite set of **nonterminals/variables**

(2) Σ is a finite set of **terminals**, and $V \cap \Sigma = \emptyset$
(a symbol is **either** a terminal **or** a nonterminal)

(3) $R \subseteq V \times (V \cup \Sigma)^*$ is a finite set of **rules/productions**

(4) $S \in V$ is **start symbol**

A rule or production is a pair

$$(A, \alpha) \in V \times (V \cup \Sigma)^*$$

also written as $A \longrightarrow \alpha$.

Ex. $G = (\underbrace{\{S\}}_{V}, \underbrace{\{0,1,\#\}}_{\Sigma}, \underbrace{\{S \to 0S1 \mid \#\}}_{R}, S)$ □

Let $G = (V, \Sigma, R, S)$ be a CFG.

Define $\underset{G}{\Longrightarrow}$ and $\overset{*}{\underset{G}{\Longrightarrow}}$ as follows

(1) For $A \to \alpha \in R$, $\beta, \gamma \in (V \cup \Sigma)^*$

we write: $\beta A \gamma \underset{G}{\Longrightarrow} \beta \alpha \gamma$

$\beta A \gamma$ directly derives $\beta \alpha \gamma$

(or directly yields)

e.g. $0^k \underbrace{S}_{A} 1^k \underset{G}{\Longrightarrow} 0^k \underbrace{\#}_{\alpha} 1^k$

(2) For $\theta_0, \ldots, \theta_m \in (V \cup \Sigma)^*$, $m \geq 0$ and

$$\theta_0 \underset{G}{\Longrightarrow} \theta_1 \underset{G}{\Longrightarrow} \cdots \underset{G}{\Longrightarrow} \theta_m \quad (\S)$$

we write $\theta_0 \overset{*}{\underset{G}{\Longrightarrow}} \theta_m$

θ_0 derives θ_m

(or yields)

($\S$) is called a derivation and m is its length

Thus, $\alpha \overset{*}{\underset{G}{\Longrightarrow}} \beta$ if and only if there exists a derivation

$$\alpha = \theta_0 \underset{G}{\Longrightarrow} \theta_1 \underset{G}{\Longrightarrow} \cdots \underset{G}{\Longrightarrow} \theta_m = \beta$$

(3) If $\alpha = \theta_0 \underset{G}{\Longrightarrow} \theta_1 \underset{G}{\Longrightarrow} \cdots \underset{G}{\Longrightarrow} \theta_i = \beta$ is a derivation of length i, we write

$$\alpha \overset{i}{\underset{G}{\Longrightarrow}} \beta$$

Remark. If G is understood, we write $\Longrightarrow$, $\overset{*}{\Longrightarrow}$ and $\overset{i}{\Longrightarrow}$.

(Note that as bin. rel. over $(V \cup \Sigma)^*$ $\overset{*}{\Longrightarrow}$ is the refl. & trans. closure of $\Longrightarrow$, and $\overset{i}{\Longrightarrow}$ is the i-fold composition of $\Longrightarrow$.) □

If $S \overset{*}{\Longrightarrow} \alpha$, then α is called a sentential form.

If $\alpha \in \Sigma^*$, i.e. α is also a term. string, then α is a sentence

The language generated by G is

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

= set of sentences generated by G.

Two CFGs G_1 and G_2 are said to be <u>equivalent</u>, $G_1 \sim G_2$, if $L(G_1) = L(G_2)$.

A language $L \subseteq \Sigma^*$ is called a <u>context. free language</u> if $L = L(G)$ for some CFG G. (CFL)

<u>Remark</u>. If $A \to \alpha_1, \ldots, A \to \alpha_k \in R$ are productions, we use the shorthand: $A \to \alpha_1 | \ldots | \alpha_k$.

<u>Ex.</u> (1) G: $S \to aSa \mid aBa$
$B \to bB \mid b$

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

e.g. $S \Rightarrow aSa$
$\vdots$
$\Rightarrow a^k S a^k \quad k \geq 0$
$\Rightarrow a^{k+1} B a^{k+1}$
$\vdots$
$\Rightarrow a^{k+1} b^l B a^{k+1} \quad l \geq 0$
$\Rightarrow a^{k+1} b^{l+1} a^{k+1}$ □

(2) G_1: $S \to AB$
$A \to aA \mid a$
$B \to bB \mid \varepsilon$

G_2: $S \to aS \mid aB$
$B \to bB \mid \varepsilon$

$L(G_1) = a^+ b^* = L(G_2)$ □
$G_1 \sim G_2$

(3) $G: \quad S \to aSd^2 \mid A$

$A \to bAc \mid bc$

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

(4) $G: \quad S \to aSa \mid bSb \mid a \mid b \mid \varepsilon$

(this is from the rec. def of palindromes)

$L(G) = \{ w \in \{a,b\}^* \mid w = w^R \}$

(5) $G: \quad S \to (S) \mid SS \mid \varepsilon$

$L(G) = D$ = set of well-formed parentheses strings over $\{(,)\}$

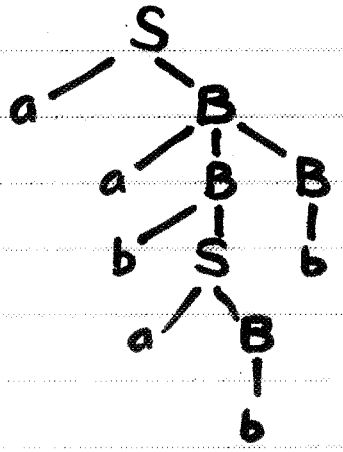
4.3. Derivation Trees

Consider the CFG G:

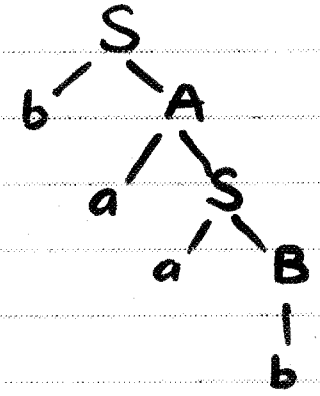
$S \to aB \mid bA$

$A \to aS \mid bAA \mid a$

$B \to bS \mid aBB \mid b$



$w = aababb$



$w = baab$

Let $G = (V, \Sigma, R, S)$ be a CFG.

A labelled, ordered tree T is called a **derivation tree** (**parse tree**) in G if:

(1) root of T is labeled by some nonterminal in G (spec. S)

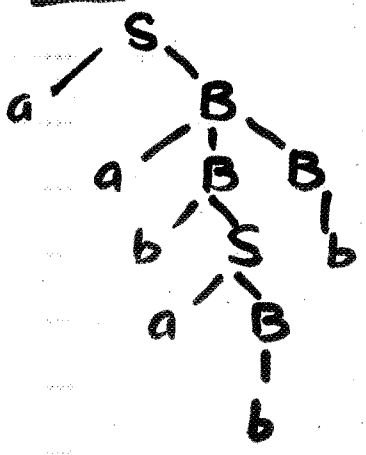
(2) if a node labeled $A \in V$ has children labeled $Y_1, \ldots, Y_k$, then

$$A \longrightarrow Y_1 \ldots Y_k \in R$$

The **frontier** of T is the string obtained by concatenating the labels of the leaves of T in the order from left to right.

If frontier of T is a terminal string then T is a **terminal derivation tree**

Ex:



$S \Rightarrow aB$
$\Rightarrow aaBB$
$\Rightarrow aaBb$
$\Rightarrow aabSb$
$\Rightarrow aababBb$
$\Rightarrow aababb$

$S \underset{lm}{\Rightarrow} aB$
$\underset{lm}{\Rightarrow} aaBB$
$\underset{lm}{\Rightarrow} aabSB$
$\underset{lm}{\Rightarrow} aababB$
$\underset{lm}{\Rightarrow} aababb$

A derivation $\theta_0 \overset{*}{\Rightarrow} \theta_m$ is said to be **leftmost**, written $\theta_0 \overset{*}{\underset{lm}{\Rightarrow}} \theta_m$, if for $i = 0, \ldots, m-1$, the production in the i-th step $\theta_{i-1} \Rightarrow \theta_i$ is applied at the leftmost nonterm. in θ_{i-1}.

Similarly we define a **rightmost** derivation.

Proposition. $S \overset{*}{\Rightarrow} \alpha$ iff there is a derivation tree with frontier α □

Remark. There is a one-one correspondence between derivation trees and leftmost (rightmost) derivations.

4.4. Designing CFGs

Although there is no recipe for this, in many cases we can apply the techniques of recursion and decomposition:

(1) Decompose the given language into simpler components

(2) Provide a recursive definition for a given language.

Ex. How to generate $L = \{a^n b^n \mid n \geq 0\}$

Observe L can be def. rec.:

(1) Basis. $\varepsilon \in L$

(2) Rec. Step. $w \in L \Rightarrow awb \in L$

(3) Nothingelse is in L.

From this we obtain the CFG

$G: \quad S \longrightarrow aSb \mid \varepsilon$

(rec. step) (basis)

Ex. How to generate

$L = \{ w \in \{a,b\}^* \mid \#_a(w) = \#_b(w) \}$

Goal: A rec. def. for L.

(1) Basis. $\varepsilon \in L$

(2) Rec. Step. If $x, y \in L$, then so are axb, bxa and xy

(3) Nothing else is in L

Q. Is this def. correct?

(Does this def. generate all strings in L and only strings in L?)

Assuming this def. is correct, we have the following CFG for L:

$G: S \to aSb \mid bSa \mid SS \mid \varepsilon$

Ex. Constr. a CFG for

$$L = \{ a^i b^j \mid i < j < 2i \}$$

Goal: To come up with a rec. def. for L.

What are the shortest strings in L?

$i = 1$: $1 < j < 2 \Rightarrow$ no value exists for j

$i = 2$: $2 < j < 4 \Rightarrow j = 3$

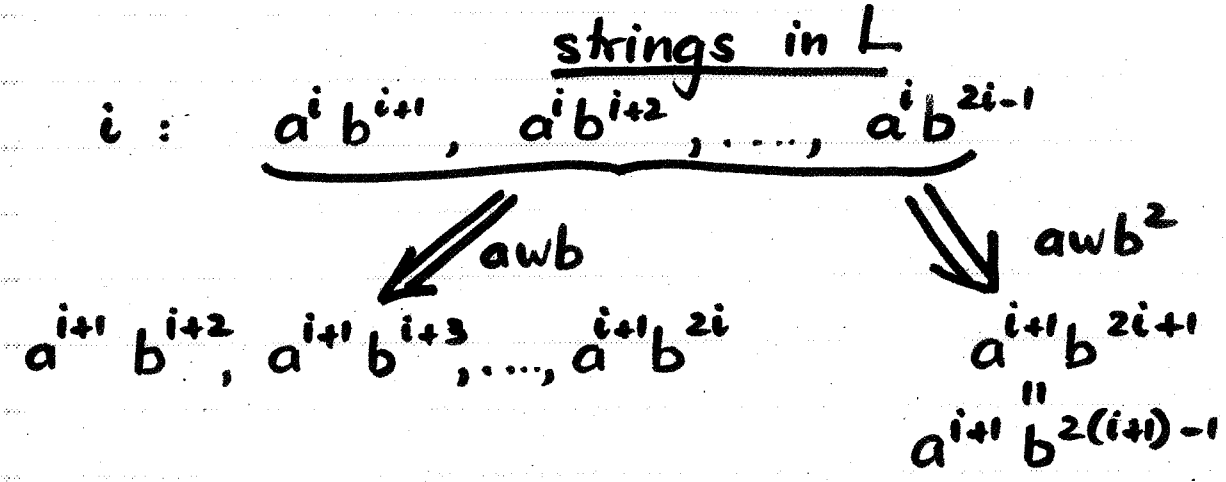
Thus the shortest string in L is a^2b^3 which should belong to basis.

Q: How to generate from a^2b^3 other strings in L?

In particular we got to generate for $i=3$ the strings a^3b^4, a^3b^5

If we use the scheme awb we obtain only a^3b^4 from a^2b^3.

It appears that we also need the scheme awb^2.



Thus, we def. L rec. as follows:

(1) Basis. $a^2b^3 \in L$

(2) Rec. Step. if $w \in L$, then so are awb and awb^2

(3) Nothing else is in L.

Hence, the CFG for L is:

$$G: \quad S \to aSb \mid aSb^2 \mid a^2b^3$$

Q. Can we provide a rigorous proof for the correctness of G ?

Ex: (Decomposition)

How to constr. a CFG for

$$L = \{ a^i b^j \mid i \neq j \} \; ?$$

Observe that

$$L = \underbrace{\{ a^i b^j \mid i < j \}}_{L_1} \cup \underbrace{\{ a^i b^j \mid i > j \}}_{L_2}$$

For L_1: $B \to aBb \quad B \to Bb \mid b$

For L_2: $A \to aAb \quad A \to aA \mid a$

Combining these productions we obtain for L:

G: $S \to A \mid B$

$A \to aAb \mid aA \mid a$

$B \to aBb \mid Bb \mid b$

Ex: Constr. a CFG for

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

Observe that

$$\hat{L} = \underbrace{\{ a^i b^j c^k \mid i \neq j \}}_{\hat{L}_1} \cup \underbrace{\{ a^i b^j c^k \mid j \neq k \}}_{\hat{L}_2}$$

Now, $\hat{L}_1 = L \cdot c^*$. For $\hat{L}_1$ we have:

$\hat{S}_1 \to S_1 C_1 \qquad C_1 \to cC_1 \mid \varepsilon$

$S_1 \to A_1 \mid B_1$

$A_1 \to aA_1b \mid aA_1 \mid a$

$B_1 \to aB_1b \mid B_1b \mid b$

Similarly for $\hat{L}_2$ we have:

$$\hat{L}_2 = a^* \{ b^j c^k \mid j \neq k \}$$

Hence, $\hat{L}_2$ can be generated by:

$$\hat{S}_2 \to A_2 S_2 \qquad A_2 \to aA_2 \mid \varepsilon$$
$$S_2 \to B_2 \mid C_2$$
$$B_2 \to bB_2c \mid bB_2 \mid b$$
$$C_2 \to bC_2c \mid C_2c \mid c$$

Adding $\hat{S} \to \hat{S}_1 \mid \hat{S}_2$ we obtain the following CFG for $\hat{L}$:

$$\hat{S} \to \hat{S}_1 \mid \hat{S}_2 \qquad \hat{S}_1 \to S_1C_1 \qquad \hat{S}_2 \to A_2S_2$$

$$\begin{cases} S_1 \to A_1 \mid B_1 \\ A_1 \to aA_1b \mid aA_1 \mid a \\ B_1 \to aB_1b \mid B_1b \mid b \end{cases} \qquad C_1 \to cC_1 \mid \varepsilon$$

$$\begin{cases} S_2 \to B_2 \mid C_2 \\ B_2 \to bB_2c \mid bB_2 \mid b \\ C_2 \to bC_2c \mid C_2c \mid c \end{cases} \qquad A_2 \to aA_2 \mid \varepsilon$$

where $\hat{S}$ is the start symbol,

and $V = \{ \hat{S}, \hat{S}_1, \hat{S}_2, S_1, S_2, A_1, B_1, C_1, A_2, B_2, C_2 \}$

4.6. The correctness problem.

Consider again the CFG G:

$$S \to aB \mid bA$$
$$A \to aS \mid bAA \mid a$$
$$B \to bS \mid aBB \mid b$$

Claim. $L(G) = \{w \in \{a,b\}^+ \mid \#_a(w) = \#_b(w)\}$
$(= L_=)$

Notation: $L_a = \{w \in \{a,b\}^+ \mid \#_a(w) = \#_b(w)+1\}$
$L_b = \{w \in \{a,b\}^+ \mid \#_b(w) = \#_a(w)+1\}$

To prove Claim we have to show the following equivalences:

(1) $A \overset{*}{\Rightarrow} w$ iff $w \in L_a$
(2) $B \overset{*}{\Rightarrow} w$ iff $w \in L_b$
(3) $S \overset{*}{\Rightarrow} w$ iff $w \in L_=$

Pf. "$\Rightarrow$" (the only-if direction)
By induction on length of derivations

Basis. (1) $A \overset{*}{\Rightarrow} w$ is of length 1. Then it's $A \Rightarrow a = w$. Obviously, $a \in L_a$

(2) $B \overset{*}{\Rightarrow} w$ is of length 1.
Then it's $B \Rightarrow b$, and $b \in L_b$

(3) $S \overset{*}{\Rightarrow} w$ is of length 2.
Then it's $S \Rightarrow aB \Rightarrow ab = w$, or
$S \Rightarrow bA \Rightarrow ba = w$. Clearly, $w \in L_=$

Inductive step.

Ind. Hypothesis: Suppose for some odd $m \geq 1$ that

(1) $A \overset{n}{\Rightarrow} w$ implies $w \in L_a$

(2) $B \overset{n}{\Rightarrow} w$ implies $w \in L_b$

(3) $S \overset{n+1}{\Rightarrow} w$ implies $w \in L_=$

for all odd $1 \leq n \leq m$.

(Note that derivations in (1)+(2) have odd length, whereas those in (3) have even length.)

To (1): Consider a longer deriv.

$$A \overset{m+2}{\Rightarrow} w.$$

There are 2 cases for the first step in this deriv.

(1.1) $A \Rightarrow aS \overset{m+1}{\Rightarrow} ay = w$

Since $S \overset{m+1}{\Rightarrow} y$ is of length $m+1$, by ind. hyp. we have $y \in L_=$

Thus, $w \in L_a$

(1.2) $A \Rightarrow bAA \overset{n_1}{\Rightarrow} bxA \overset{n_2}{\Rightarrow} bxy$ (with $bxy = w$)

Since $A \overset{n_1}{\Rightarrow} x$, $A \overset{n_2}{\Rightarrow} y$, $n_1, n_2 \leq m$, by ind. hyp., $x, y \in L_a$

Thus, $w \in L_a$

To (2): Consider a longer derivation

$B \overset{m+2}{\Longrightarrow} w$

As in (1), by similar argument, we obtain $w \in L_b$

To (3): Consider the derivation

$S \overset{m+3}{\Longrightarrow} w.$

For the first step in $S \overset{m+3}{\Longrightarrow}$ there are 2 possibilities:

(3.1) $S \Longrightarrow aB \overset{m+2}{\Longrightarrow} ay = w.$

As proved in (2), $B \overset{m+2}{\Longrightarrow} y$ implies that $y \in L_b$. Therefore,

$w \in L_=$

(3.2) $S \Longrightarrow bA \overset{m+2}{\Longrightarrow} ax = w$

By similar arg. as in (3.1), we have $w \in L_=$

This concludes the ind. step, and hence "$\Longrightarrow$" is proved.

We now prove "$\Longleftarrow$"

"$\Longleftarrow$": We show that

(1) $w \in L_a$ implies $A \overset{*}{\Longrightarrow} w$

(2) $w \in L_b$ " $B \overset{*}{\Longrightarrow} w$

(3) $w \in L_=$ " $S \overset{*}{\Longrightarrow} w$

By induction on $|w|$

Basis.

(1) $w = a$: Clearly $A \Rightarrow a$

(2) $w = b$: $B \Rightarrow b$

(3) $w = ab$ or $w = ba$.

Clearly, $S \Rightarrow aB \Rightarrow ab$

and $S \Rightarrow bA \Rightarrow ba$

Inductive Step.

Ind. Hyp. Suppose for some odd $m \geq 1$ that

(1) $w \in L_a \wedge |w| = n \Rightarrow A \overset{*}{\Rightarrow} w$

(2) $w \in L_b \wedge |w| = n \Rightarrow B \overset{*}{\Rightarrow} w$

(3) $w \in L_= \wedge |w| = n+1 \Rightarrow S \overset{*}{\Rightarrow} w$

for all odd $1 \leq n \leq m$.

To (1): Let $w \in L_a$ and $|w| = m+2$.

w begins with a or with b

(1.1) $w = bx$. Then $|x| = m+1$ and

$$\#_a(x) = \#_b(x) + 2.$$

Q. How do we generate bx from A ?

It got to be of the form

$$A \Rightarrow bAA \overset{*}{\Rightarrow} bx,$$

i.e., $AA \overset{*}{\Rightarrow} yA \overset{*}{\Rightarrow} yz = x$

If this is to be achieved, then one should be able to write x as $x = yz$ and $y, z \in L_a$

Indeed one must be able to prove the following fact:

<u>Fact</u>. If $\#_a(x) = \#_b(x) + 2$, then x can be written as $x = yz$ s.t. $y, z \in L_a$

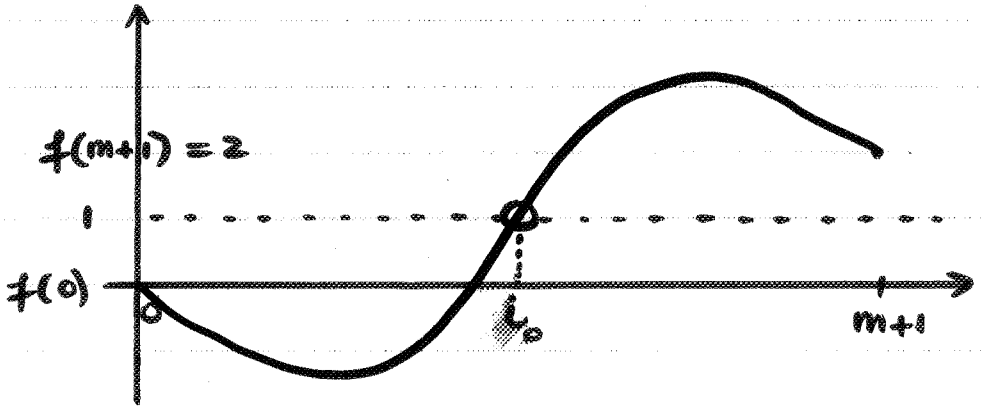
<u>Pf of Fact</u>. Let $x = x_1 \dots x_{m+1}$.

Define $f : \{0, \dots, m+1\} \longrightarrow \mathbb{Z}$

by $f(i) := \#_a(x_1 \dots x_i) - \#_b(x_1 \dots x_i)$

Then: $f(0) = 0$, $f(m+1) = 2$

and the graph of f has the form:



So there exists $0 < i_0 < m+1$

s.t. $f(i_0) = 1$

Letting $y = x_1 \cdots x_{i_0}$

and $z = x_{i_0+1} \cdots x_{m+1}$

we have: $y, z \in L_a \wedge x = yz$ □

Now, both $y, z \in L_a$ and $|y|, |z| \leq m$,

so by ind. hyp.

$$A \overset{*}{\Rightarrow} y \quad \text{and} \quad A \overset{*}{\Rightarrow} z.$$

Thus we have:

$$A \Rightarrow bAA \overset{*}{\Rightarrow} byA \overset{*}{\Rightarrow} byz = w$$

(1.2) $w = ax$. Then $x \in L_=$ and $|x| = m+1$

By ind. hyp. $S \overset{*}{\Rightarrow} x$.

Thus, $A \Rightarrow aS \overset{*}{\Rightarrow} ax = w$.

To (2). Similar to (1).

To (3) Let $w \in L_=$ and $|w| = m+3$

(3.1) $w = ax$: Then $x \in L_b$, $|x| = m+2$.

From (2) it follows that $B \overset{*}{\Rightarrow} x$.

Thus, $S \Rightarrow aB \overset{*}{\Rightarrow} ax = w$

(3.2) $w = bx$: Then $x \in L_a$, $|x| = m+2$.

From (1) it follows that $A \overset{*}{\Rightarrow} x$.

Thus, $S \Rightarrow bA \overset{*}{\Rightarrow} bx = w$.

This completes the proof of Claim.

□□

4.7. Regular Grammars

Q: Can we def. a subclass of CFGs that generate the reg. languages?

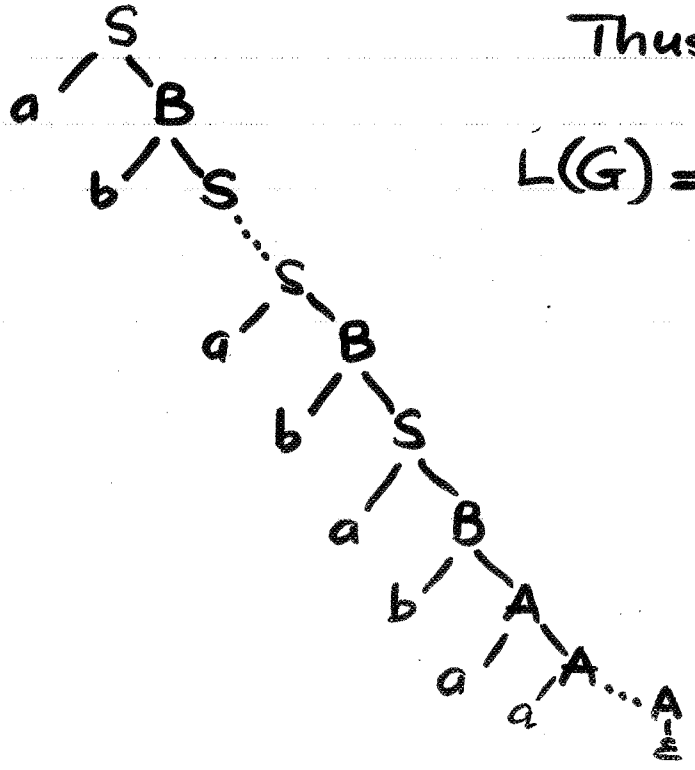
Def. A **regular grammar** is a CFG in which productions are of form:

(1) $A \longrightarrow aB$
(2) $A \longrightarrow a$
(3) $A \longrightarrow \varepsilon$

where $A, B \in V$ and $a \in \Sigma$.

Ex: (1) G: $S \longrightarrow aB \mid \varepsilon$
$B \longrightarrow bS \mid bA$
$A \longrightarrow aA \mid \varepsilon$

A derivation tree has the form



Thus

$$L(G) = (ab)^{+} a^{*} + \varepsilon$$

$$\neq (ab)^{*} a^{*}$$

(2) G_1: $S \to AB$, $A \to aA \mid a$, $B \to bB \mid \varepsilon$ $\qquad$ G_2: $S \to aS \mid aB$, $B \to bB \mid \varepsilon$

$L(G_1) = a^+b^* = L(G_2)$

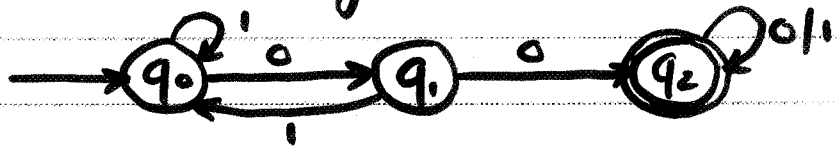
(Note that G_2 is reg. whereas G_1 is not.) □

Regular grammars & Regular sets.

Goal: To show reg. grammars $\Leftrightarrow$ FAs

Proposition. $L = L(M)$ for some DFA M $\Rightarrow$ $L = L(G)$ for some reg. gr. G.

Pf: Consider e.g. the DFA M:



Q: How to constr. equiv. reg. gr. G?

Idea: Productions $A \to aB$ in G should simulate trans.  in M.

Thus, G contains productions:

$q_0 \to 1q_0$ $\qquad$ $q_2 \to 0q_2$

$q_0 \to 0q_1$ $\qquad$ $q_2 \to 1q_2$

$q_1 \to 1q_0$

$q_1 \to 0q_2$

As q_2 is final, computation may end and accepts at q_2. Hence in G we have the prod. $q_2 \to \varepsilon$

Thus the computation

$$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_2 \xrightarrow{0} (q_2)$$

corresponds to the derivation

$$q_0 \Rightarrow 1q_0 \Rightarrow 10q_1 \Rightarrow 100q_2 \Rightarrow 1001q_2 \Rightarrow 10010q_2 \Rightarrow 10010$$

Given DFA $M = (Q, \Sigma, \delta, q_0, F)$ the reg. grammar $G = (V, \Sigma, R, S)$ is constr. as follows:

(1) $V = Q$

(2) $S = q_0$

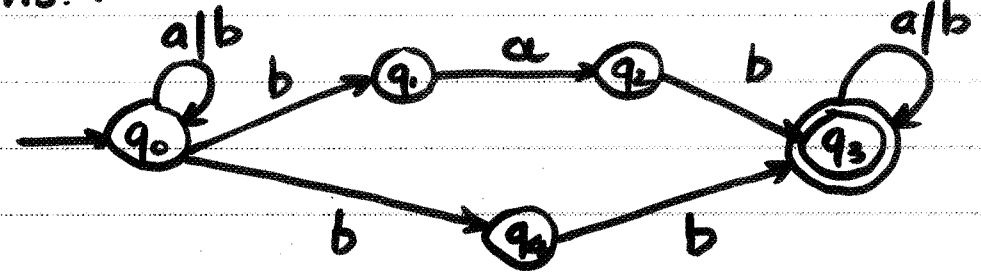
(3) R contains the productions:

- $\delta(q,a) = q' \iff q \to aq' \in R$
- for each $q_f \in F$:

$$q_f \to \varepsilon \in R.$$

Then $L(M) = L(G)$ □

Ex: Consider an NFA M without ε-trans.:



The equiv. reg. grammar G is:

$q_0 \longrightarrow aq_0 \mid bq_0 \mid bq_1 \mid bq_4$

$q_1 \longrightarrow aq_2$

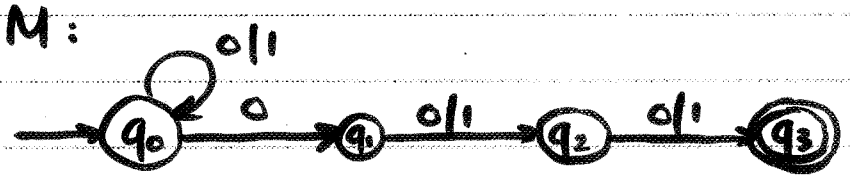
$q_2 \longrightarrow bq_3$

$q_4 \longrightarrow bq_3$

$q_3 \longrightarrow aq_3 \mid bq_3 \mid \varepsilon$

(Note the above const. works for NFAs without ε-transitions as well.) □

Ex: M:



G:

$q_0 \longrightarrow 0q_0 \mid 1q_0 \mid 0q_1$

$q_1 \longrightarrow 0q_2 \mid 1q_2$

$q_2 \longrightarrow 0q_3 \mid 1q_3$

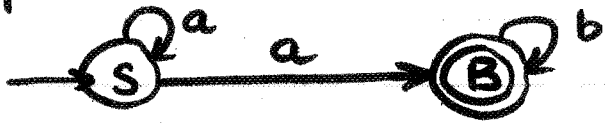
$q_3 \longrightarrow \varepsilon$ □

Proposition. $L = L(G)$ for a reg. gr. G $\Longrightarrow$ $L = L(N)$ for some NFA N.

Pf. The idea is similar to previous const.
Consider e.g. reg. grammar G:

$$S \to aS \mid aB$$
$$B \to bB \mid \varepsilon$$

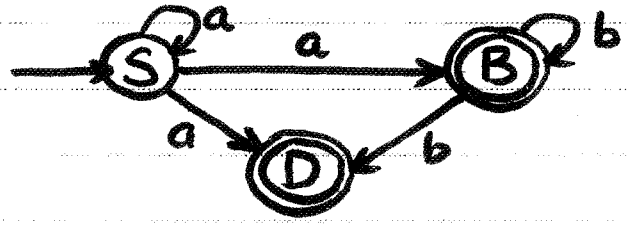
The equiv. NFA N is:



Suppose that G also contains

$$S \to a \quad \text{and} \quad B \to b$$

To simulate these productions we introduce an additional final state D:



Thus, given $G = (V, \Sigma, R, S)$ the equiv. NFA $N = (Q, \Sigma, \delta, q_0, F)$ is:

(1) $Q = V \cup \{D\}$ (2) $q_0 = S$

(3) $F = \{A \mid A \to \varepsilon \in R\} \cup \{D\}$

(4) δ is def by:

$\delta(A, a) = B$ iff $A \to aB \in R$

$\delta(A, a) = D$ iff $A \to a \in R$

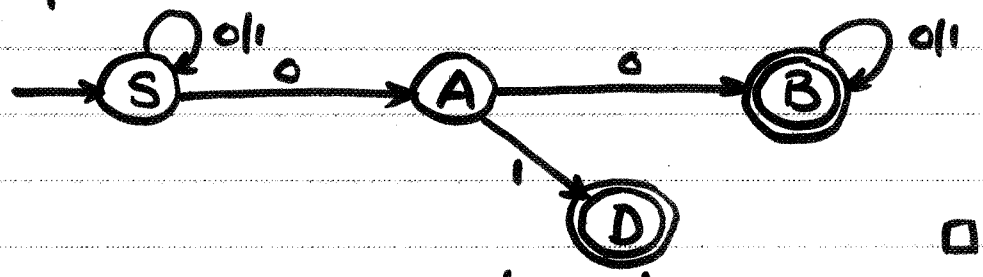
□

Ex: G: $S \to 0S \mid 1S \mid 0A$

$A \to 0B \mid 1$

$B \to 0B \mid 1B \mid \varepsilon$

The equiv. NFA is:



□

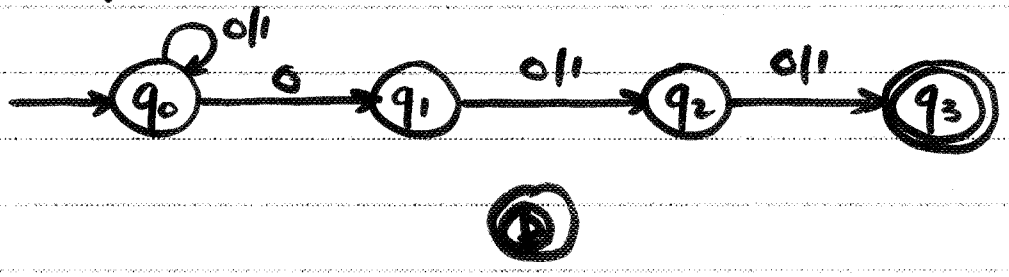
Ex: G: $q_0 \to 0q_0 \mid 1q_0 \mid 0q_1$

$q_1 \to 0q_2 \mid 1q_2$

$q_2 \to 0q_3 \mid 1q_3$

$q_3 \to \varepsilon$

The equiv. NFA is:



□