PART II: Chapter 2

Linear Grammars and Normal Forms

Linear Grammar

G = (N,Σ,S,P) : a CFG

- A,B: nonterminals
- a: terminal symbol
- $y \in \Sigma^*$, $x \in \Sigma^*$.

Grammar Type	Production form
right linear	A → yB or A → x
Strongly right linear	A → aB | B | ε
Left linear	A→ By or A → x
Strongly left linear	A → Ba | B | ε

- Notes:
  - 1. All types of linear grammars are CFGs.
  - 2. All types of linear grammars generate the same class of languages (i.e., regular languages)

Theorem: For any language L: the following statements are equivalent:

- 0. L is regular
- 1. L = L(G1) for some RG G1 2. L=L(G2) for some SRG G2
- 3. L=L(G3) from some LG G3 4. L=L(G4) for some SLG G4

Equivalence of linear languages and regular sets

- **Pf: (2) => (1) and (4)=>(3) : trivial since SRG (SLG) are special kinds of RG (LG).**
- **(1)=>(2) :1. replace each rule of the form:**

  $$A \rightarrow a_1\ a_2 \ldots a_n\ B \quad (n > 1)$$

  by the following rules

  $$A \rightarrow a_1\ B_1,\ \ B_1 \rightarrow a_2\ B_2,\ \ldots,\ B_{n-2} \rightarrow a_{n-1}\ B_{n-1},\ B_{n-1} \rightarrow a_n\ B$$

  where $B_1, B_2, \ldots, B_{n-1}$ are new nonterminal symbols.

  2. Replace each rule of the form:

  $$A \rightarrow a_1\ a_2 \ldots a_n \quad (n \geq 1)$$

  by the following rules

  $$A \rightarrow a_1B_1\ ,\ \ B_1 \rightarrow a_2B_2,\ \ldots,\ B_{n-1} \rightarrow a_nB_n,\ \ B_n \rightarrow \varepsilon$$

  3. Let G' be the resulting grammar. Then L(G) = L(G').
- **(3)=>(4) : Similar to (1) =>(2).**

$A \rightarrow B\ a_1\ a_2 \ldots a_n \quad (n > 1) \Longrightarrow A \rightarrow B_na_n,\ \ B_n \rightarrow B_{n-1}a_{n-1},\ \ldots,\ B_2 \rightarrow Ba_1$

$A \rightarrow a_1\ a_2 \ldots a_n \quad (n \geq 1) \Longrightarrow A \rightarrow B_na_n,\ \ B_n \rightarrow B_{n-1}a_{n-1},\ \ldots,\ B_2 \rightarrow B_1a_1\ ,\ B_1 \rightarrow \varepsilon$

Example:

- The right linear grammar :
  - S → abab S and S → abc
  - can be converted into a SRG as follows:
  - S → ababS =>
  - S → a [babS]
  - [babS] → b [abS]
  - [abS] → a [bS]
  - [bS] →b S
  - S → abc =>
  - S → a [bc]
  - [bc] → b [c]
  - [c] → c []
  - [] → ε

RGs and FAs

- pf: (0) =>(2), (0)=>(4)
- Let $M = (Q,\Sigma,\delta,S,F)$: A NFA allowing empty transitions.
- Define a SRG G_2 and a SLG G_4 as follows:
- $G_2 = (N_2, \Sigma, S_2, P_2)$ $G_4 = (N_4, \Sigma, S_4, P_4)$ where
  - 1. $N_2 = Q \cup \{S_2\}$, $N_4 = Q \cup \{S_4\}$, where S_2 and S_4 are new symbols and
  - $P_2 = \{S_2 \rightarrow A \mid A \in S\} \cup \{A \rightarrow aB \mid B \in \delta(A,a)\}$ $\cup \{A \rightarrow \varepsilon \mid A \in F\}$. // to go to a final state from A, use 'a' to reach B and then from B go to a final state.
  - $P_4 = \{S_4 \rightarrow A \mid A \in F\} \cup \{B \rightarrow Aa \mid B \in \delta(A,a)\}$ $\cup \{A \rightarrow \varepsilon \mid A \in S\}$. // to reach B from a start state, reach A from a start state and then consume a.

- Lem 01: If $S_2 \rightarrow^{+}_{G2} \alpha \notin \Sigma^*$, then α = xB where $x \in \Sigma^*$ and $B \in Q$
- Lemma 1: $S_2 \rightarrow^{+}_{G2} xB$ iff $B \in \Delta(S,x)$.

--- can be proved by ind. on derivation length(=>) and x (<=).

Hence $x \in L(G_2)$

iff $S_2 \rightarrow^{*}_{G2} x$ iff $S_2 \rightarrow^{+}_{G2} xB \rightarrow_{G2} x$ for a $B \in F$.

iff $B \in \Delta(S,x)$ and $B \in F$ iff $x \in L(M)$

- Lem 02: If $S_4 \rightarrow^{+}_{G4} \alpha \notin \Sigma^*$, then α=Bx where $x \in \Sigma^*$ and $B \in Q$.
- Lemma 2: $S_4 \rightarrow^{+}_{G4} Bx$ iff $F \cap \Delta(B,x) \neq \emptyset$.

Hence $S_4 \rightarrow^{*}_{G4} x$

iff $S_4 \rightarrow^{*}_{G4} Bx \rightarrow_{G4} x$ for some start state B

iff $B \in S$ and $F \cap \Delta(B,x) \neq \emptyset$ iff $x \in L(M)$

Theorem: $L(M) = L(G_2) = L(G_4)$.

From FA to LGs: An example

- Let M = ({A,B,C,D}, {a,b}, δ, {A,B},{B,D}) where
- δ is given as follows:

```
> A  <-- a --> C
  ^            ^
  b            b
  V            V
>(B) <-- a-->  (D)
```

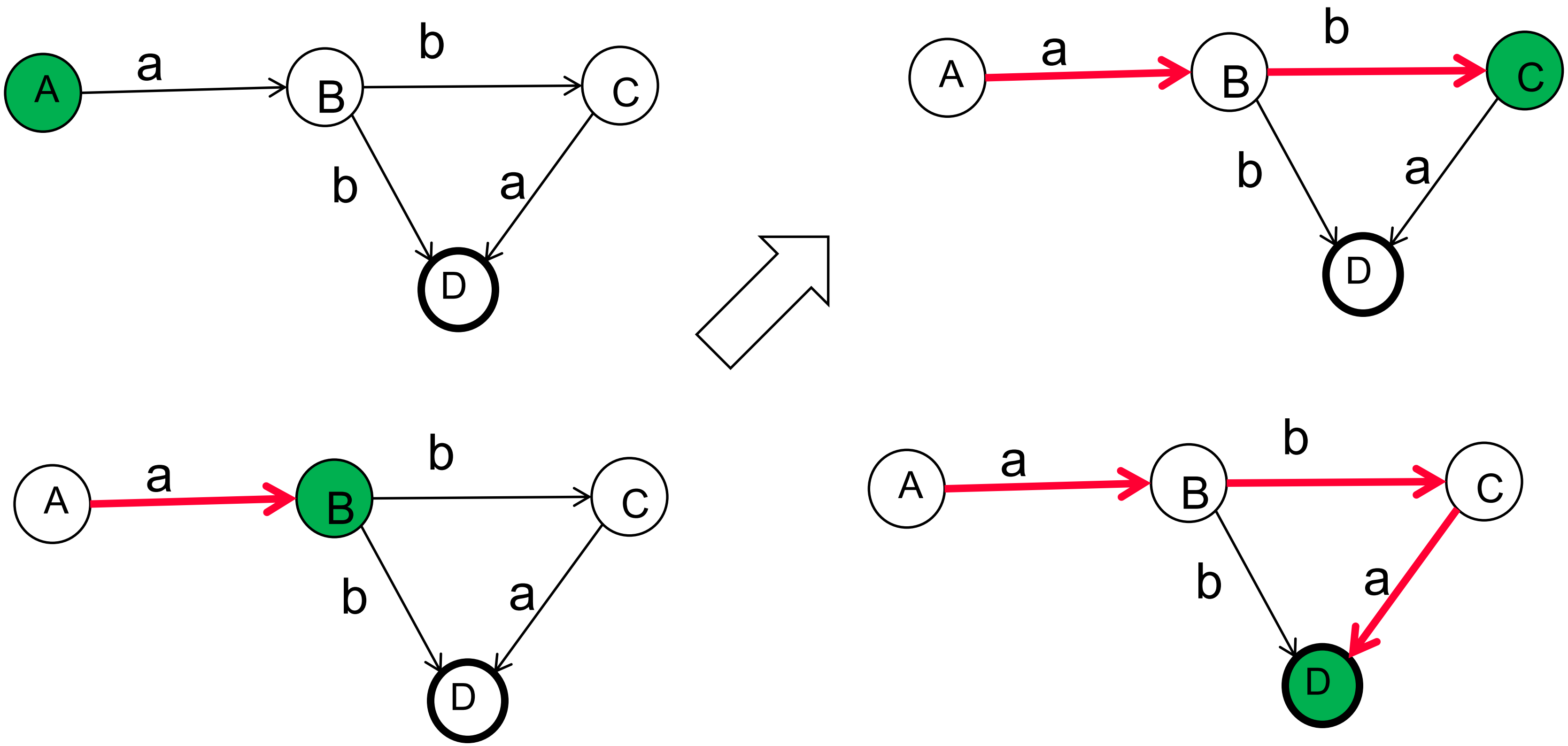
==> G2 = ? G4 = ?

E –a –> F is translated to :

1. (G2) E →aF : E → ε // if E is a final state
   - To reach a final state from E, go to F first by consuming an 'a' and then try to reach a final state from F.
2. (G4) F → Ea : E →ε // if E is a start state
   - How to reach F from a start state? go to E first and then by consuming a, you can reach F.

Motivation: Derivation and path walk

- S→ A → aB → abC →abaD → aba.
  => { A →aB, B →bC, C →aD, D → ε … }

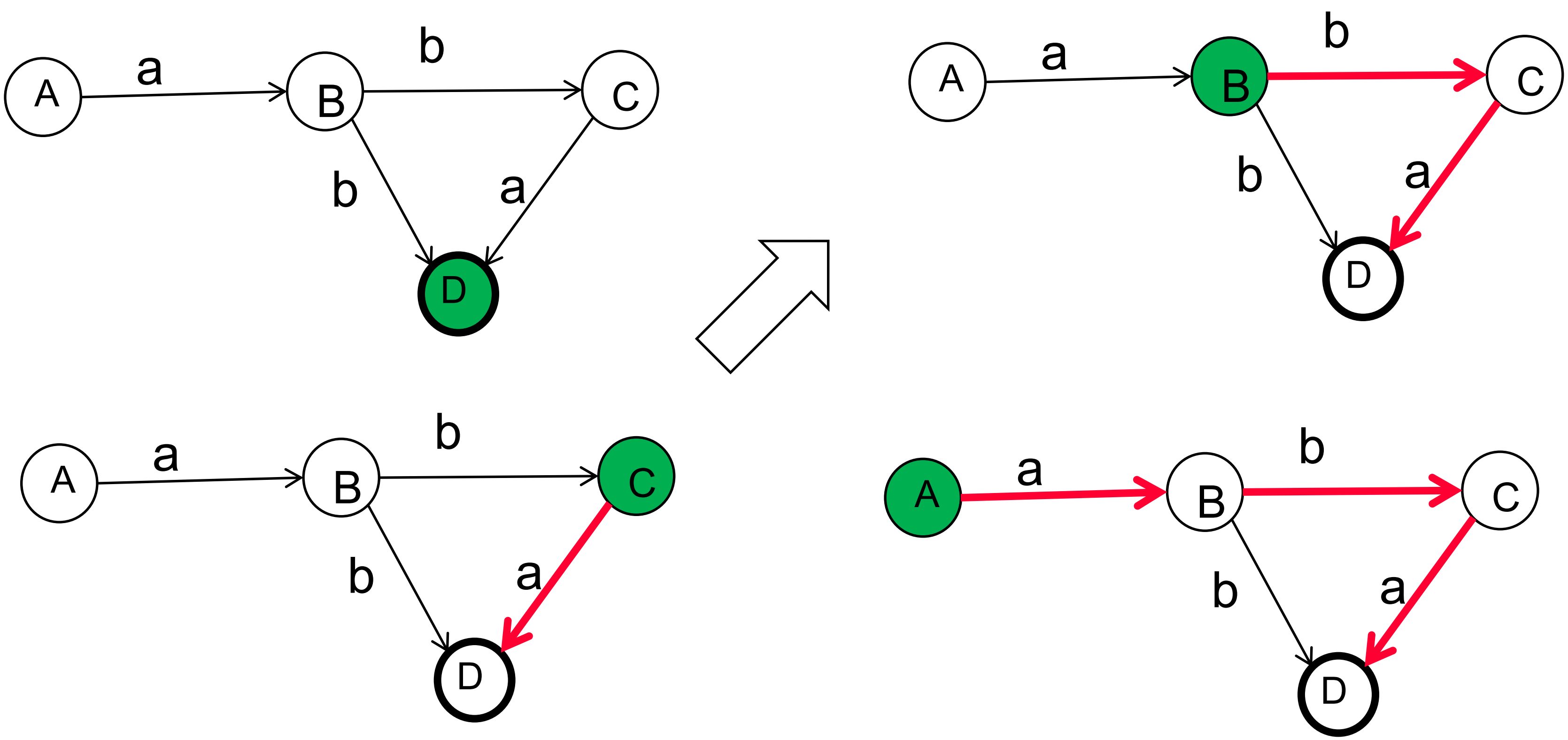


Conclusion: The forward walk of a path from a start state to a final state is the same as the derivation of a SRG grammar.

Derivation and backward path walk

- **S→D → Ca → Bba →Aaba → aba.**

  => { D →Ca, C →Bb, B →Aa, A → ε … }



Conclusion: The backward walk of a path from a start state to a final state is the same as the derivation of a SLG grammar.

From FA to LGs: an example

- Let M = ({A,B,C,D}, {a,b}, δ, {A,B},{B,D}) where
- δ is given as follows:

```
> A  <-- a --> C
  ^            ^
  b            b
  V            V
>(B) <-- a-->  (D)
```

==> G2 = ?

sol: S2 → A | B

A → aC | bB

B → aD | bA | ε

C → aA | bD

D → aB | bC | ε

G4 = ?

sol: S4 → B | D

B → Ab | Da | ε

D → Cb | Ba

C → Aa | Db

A → Bb | Ca | ε

From Linear Grammars to FAs

- G = (N,Σ,S,P) : a SRG

  Define M = (N,Σ,δ,{S},F) where
  - F = {A | A → ε ∈ P} and
  - δ = {(A,a,B) | A → aB ∈ P,
  - a ∈ Σ U {ε} }

Theorem: L(M) = L(G).

- G = (N,Σ,S,P) : a SLG

Define M' = (N,Σ,δ,S',{S}) where
  - S' = {A | A → ε ∈ P} and
  - δ = {(A,a,B) | B → Aa ∈ P,
  - a ∈ Σ U {ε} }

Theorem: L(M') = L(G).

Example:

G : S → aB | bA
B → aB |ε
A → bA |ε
=> M = ?

Example:

G: S → Ba | Ab
A → Ba |ε
B → Ab |ε
==> M' = ?

Other types of transformations

- FA ↔ LG = {SLG, SRG } (ok!)
- FA ↔ Regular Expression (ok!)
- SLGs ↔ SRGs (?)
  - SLG ↔ FA ↔ SRG
- LG ↔ Regular Expression (?)
  - LG ↔ FA ↔ Regular Expression
- Ex: Translate the SRG G: S→aA | bB, A →aS | ε, B → bA | bS | ε into an equivalent SLG.

sol: The FA corresponding to G is M = (Q, {a,b}, δ, S, {A,B}), where Q= {S,A,B} and δ = { (S, a, A), (S,b,B), (A, a,S), (B,b,A),(B,b,S)}

So the SLG for M (and G as well) is

S' → A | B, --- final states become start symbol; S' is the new start symbol

S → ε, --- start state becomes empty rule

A → Sa, B → Sb, S→Aa, A → Bb, S → Bb. // do you find the rule from SRG to SLG ?

Exercises

- **Convert the following SRG into an equivalent SLG ?**
  - **S → aS | bA | aB | ε**
  - **A → aB | bA | aS | ε**
  - **B → bA | aS**

Rules:

A → aB ➔ B →Aa

A → B ➔ B → A

empty rule: A → e ➔ S' → A

start symbol: S ➔ S → e

- **Convert the following SLG into an equivalent SRG ?**
  - **S → Ca | Ab | Ba**
  - **A → Ba | Cb | ε**
  - **B → Ab | Sa**
  - **C → Aa | Bb | ε**

Rules:

A → Ba ➔ B → aA

A → B ➔ B → A

empty rule: A → e ➔ S' → A

start symbol: S ➔ S → e

Chomsky normal form and Greibach normal form

- G = (N,Σ,P,S) : a CFG
- G is said to be in *Chomsky Normal Form (CNF)* iff all rules in P have the form:

   A → a   or   A → BC

   where a ∈ Σ and A, B,C ∈ N. Note: B and C may equal to A.
- G is said to be in Greibach Normal Form (GNF) iff all rules in P have the form:

   A → a $B_1B_2\ldots B_k$

   where $k \geq 0$, $a \in \Sigma$ and $B_i \in N$ for all $1 \leq i \leq k$.

   Note: when k = 0 => the rule reduces to A → a.

Ex: Let G_1: S → AB | AC |SS, C → SB, A → [, B →]

G_2: S → [B | [SB | [BS | SBS, B →]

==> G_1 is in CNF but not in GNF

G_2 is in GNF but not in CNF.

Remarks about CNF and GNF

1. $L(G_1) = L(G_2) = \text{PAREN} - \{\varepsilon\}$.
2. No CFG in CNF or GNF can produce the null string ε. (Why ?)

Observation: Every rule in CNF or GNF has the form $A \to \alpha$ with $|A| = 1 \le |\alpha|$ since ε can not appear on the RHS.

So

Lemma: G: a CFG in CNF or GNF. Then $\alpha \to \beta$ only if $|\alpha| \le |\beta|$.
Hence if $S \to^* x \in \Sigma^* \Rightarrow |x| \ge |S| = 1 \Rightarrow x \neq \varepsilon$.

3. Apart from (2), CNF and GNF are as general as CFGs.

Theorem 21.2: For any CFG G, $\exists$ a CFG G' in CNF and a CFG G'' in GNF s.t. $L(G') = L(G'') = L(G) - \{\varepsilon\}$.

Generality of CNF

- ε-rule: A → ε.
- unit (chain) production: A → B.
- Lemma: G: a CFG without unit and ε-rules. Then ∃ a CFG G' in CNF form s.t. L(G) = L(G').

Ex21.4: G: S → aSb | ab has no unit nor ε-rules.

==> 1. For terminal symbol a and b, create two new nonterminal symbol A and B and two new rules:

- A → a, B → b.
- 2. Replace every a and b in G by A and B respectively.
- => S → ASB | AB, A → a, B → b.
- 3. S → ASB is not in CNF yet ==> split it into smaller parts:
- (Say, let AS =) ==> S → B and → AS.
- 4. The resulting grammar :
- S → B | AB, A → a, B → b, → AS is in CNF.

generality of CNF

Ex21.5: G: S → [S]S | SS | [ ] ==>

- **A → [, B →], S → ASBS | SS | AB**
- **==> replace S → ASBS by S→ S → ASB,**
- **==>replace → ASB by → B and → AS.**
- **==> G': A →[, B →],**
- **S → S |SS |AB,**
- **→ B, → AS.**
- **(2) another possibility:**
- **S → ASBS becomes S → , → AS, → BS.**

Problem: How to get rid of ε and unit productions:

Elimination of e-rules (cont'd)

- It is possible that $S \rightarrow^* w \rightarrow w'$ with $|w'| < |w|$ because of the ε-rules.

Ex1: G: $S \rightarrow SaB \mid aB$ $B \rightarrow bB \mid \varepsilon$.

=> $S \rightarrow SaB \rightarrow SaBaB \rightarrow aBaBaB \rightarrow aaBaB \rightarrow aaaB \rightarrow aaa$.

$L(G) = (aB)^+ = (ab^*)^+$

Another equivalent CFG w/o ε-rules:

Ex2: G': $S \rightarrow SaB \mid Sa \mid aB \mid a$ $B \rightarrow bB \mid b$.

$S \rightarrow^* S\,(a + aB)^* \rightarrow (a+aB)^+$ $B \rightarrow^* b^*B \rightarrow b^+$.

=> $L(G') = L(S) = (a + ab^+)^+ = (ab^*)^+$

Problem: Is it always possible to create an equivalent CFG w/o ε-rules ?

Ans: yes! but with proviso.

Elimination of ε-rules (cont'd)

Def: 1. a nonterminal A in a CFG G is called nullable if it can derive the empty string. i.e., $A \to^* \varepsilon$.

2. A grammar is called noncontracting if the application of a rule cannot decrease the length of sentential forms.

(i.e.,for all $w,w' \in (\Sigma \cup N)^*$, if $w \to w'$ then $|w'| \geq |w|$.)

Lemma 1: G is noncontracting iff G has no ε-rule.

pf: G has ε-rule $A \to \varepsilon$ => $1 = |A| > |\varepsilon| = 0$.

G contracting => $\exists \alpha,\beta \in (N \cup \Sigma)^*$ and $A \to \varepsilon$ with $\alpha A \beta \to \alpha\beta$.

=> G contains an ε-rule.

Simultaneous derivation:

Def: G: a CFG. $==>_G$: a binary relation on $(N \cup \Sigma)^*$ defined as follows: for all $\alpha,\beta \in (N\cup\Sigma)^*$, $\alpha ==> \beta$ iff

there are $x_0,x_1,..,x_n \in \Sigma^*$, rules $A_1 \rightarrow \gamma_1, \ldots, A_n \rightarrow \gamma_n$ ($n > 0$) s.t.

$\alpha = x_0\ A_1\ x_1\ \ A_2\ \ x_2 \ldots\ \ A_n\ \ x_n$ and

$\beta = x_0\ \ \gamma_1\ x_1\ \ \gamma_2\ \ \ x_2 \ldots\ \ \gamma_n\ \ x_n$

$==>^n$ and $==>^*$ are defined similarly like $\rightarrow^n$ and $\rightarrow^*$.

Define $==>^{(n)}\ =_{def}\ (\bigcup_{k \le n} ==>^k)$.

Lemma:

1. if $\alpha ==> \beta$ then $\alpha \rightarrow^* \beta$. Hence $\alpha ==>^* \beta$ implies $\alpha \rightarrow^* \beta$.
2. If β is a terminal string, then $\alpha \rightarrow^n \beta$ implies $\alpha ==>^{(n)} \beta$.
3. $\{x \in \Sigma^* \mid S ==>^* x\} = L(G) = \{x \in \Sigma^* \mid S \rightarrow^* x\}$.

Find nullable symbols in a grammar

Problem: How to find all nullable nonterminals in a CFG ?

Note: If A is nullable then there are numbers n s.t. $A \Longrightarrow^{(n)} \varepsilon$.

Now let $N_k = \{ A \in N \mid A \Longrightarrow^{(k)} \varepsilon \}$.

1. N_G (the set of all nullable nonterminals of G) = $\bigcup_{k \geq 0} N_K$.
2. $N_1 = \{A \mid A \rightarrow \varepsilon \in P\}$.
3. $N_{k+1} = N_k \cup \{A \mid A \rightarrow X_1X_2\ldots X_n \in P$ (n >= 0) and All X_is $\in N_k \}$.

Ex: G : $S \rightarrow ACA$ $\quad A \rightarrow aAa \mid B \mid C$

$B \rightarrow bB \mid b$ $\quad C \rightarrow cC \mid \varepsilon$.

=> N_1 = ? {C}

$N_2 = N_1 \cup$?

$N_3 = N_2 \cup$?

N_G = ?

Exercises: 1. Write an algorithm to find N_G.

2. Given a CFG G, how to determine if $\varepsilon \in L(G)$?

Adding rules into grammar w/t changing language

Lem 1.4: G = (N,Σ,P,S) : a CFG s.t. $A \rightarrow^* \omega$. Then the CFG G' = (N,Σ, PU{ $A \rightarrow \omega$}, S) is equivalent to G.

pf: L(G) ⊆ L(G') : trivial since $\rightarrow_G \subseteq \rightarrow_{G'}$.

L(G') ⊆ L(G): First define $\alpha \text{ ->>}^k_{G'} \beta$ iff ($\alpha \rightarrow^*_{G'} \beta$ and the rule $A \rightarrow \omega$ was applied k times in the derivation).

Now it is easy to show by ind. on k that

if $\alpha \text{ ->>}^{k+1}_{G'} \beta$ then $\alpha \text{ ->>}^k_{G'} \beta$ (and hence $\alpha \text{ ->>}^0_{G'} \beta$ and $\alpha \rightarrow^*_G \beta$).

Hence $\alpha \rightarrow^*_{G'} \beta$ implies $\alpha \rightarrow^*_G \beta$ and L(G') ⊆ L(G).

Theorem 1.5: for any CFG G , there is a CFG G' containing no ε-rules s.t. L(G') = L(G) - {ε}.

Pf: Define G'' and G' as follows:

1. Let P'' = P U Δ where Δ = {$A \rightarrow X_0X_1\ldots X_n$ | $A \rightarrow X_0A_1X_1\ldots A_nX_n \in P$, $n \geq 1$, All A_is are nullable symbols and $X_i \in (N \cup \Sigma)^*$. }.
2. Let P' be the resulting P'' with all ε-rules removed.

Elimination of e-rules (con't)

By lem 1.4, L(G) = L(G''). We now show L(G') = L(G'') - {ε}.

1. Since P' $\subseteq$ P'', L(G') $\subseteq$ L(G''). Moreover, since G' contains no ε-rules, $\varepsilon \notin$ L(G') Hence L(G') $\subseteq$ L(G'') - {ε}.
2. For the other direction, first define $S \dashrightarrow^k_{G''} \beta$ iff

   $S \rightarrow^*_{G''} \beta$ and all ε-rules $A \rightarrow \varepsilon$ in P'' are used k times totally in the derivation. Note: if $S \dashrightarrow^0_{G''} \beta$ then $S \rightarrow^*_{G'} \beta$.

we show by induction on k that

if $S \dashrightarrow^{k+1}_{G''} \beta$ and $\beta \neq \varepsilon$ then

$S \dashrightarrow^k_{G''} \beta$ for all k ≥0 and hence $S \dashrightarrow^0_{G''} \beta$ and $S \rightarrow^*_{G'} \beta$.

As a result if $S \rightarrow^*_{G''} \beta \in \Sigma^+$ then $S \rightarrow^*_{G'} \beta$. Hence L(G'')-{ε} $\subseteq$ L(G') .

But now if $S \dashrightarrow^{k+1}_{G''} \beta$ then

$S \rightarrow^*_{G''} \mu B \nu$ --($B \rightarrow x\underline{A}y$)-→ $\mu xAy\nu \rightarrow w_1 \rightarrow \ldots \rightarrow \alpha'\underline{A}\beta'$ --($A \rightarrow \varepsilon$)

$\rightarrow \alpha'\beta' \rightarrow \ldots \rightarrow \beta$ and then

$S \rightarrow^*_{G''} \mu B \nu$ --($B \rightarrow xy$) → $\mu xy\nu \rightarrow w'_1 \rightarrow \ldots \rightarrow \alpha'\beta' \rightarrow \ldots \rightarrow \beta$.

hence $S \dashrightarrow^k_{G''} \beta$. QED

Example 1.4:

Ex 1.4: G: $S \to ACA$ $A \to aAa \mid B \mid C$

$B \to bB \mid b$ $C \to cC \mid \varepsilon$.

$\Rightarrow N_G = \{C, A, S\}$.

Hence P'' = $P \cup \{$ $S \to ACA \mid AC \mid CA \mid AA \mid A \mid C \mid \varepsilon$

$A \to aAa \mid aa \mid B \mid C \mid \varepsilon$

$B \to bB \mid b$

$C \to cC \mid c \mid \varepsilon$ $\}$

and P' = $\{$ $S \to ACA \mid AC \mid CA \mid AA \mid A \mid C$

$A \to aAa \mid aa \mid B \mid C$

$B \to bB \mid b$

$C \to cC \mid c$ $\}$

Elimination of unit-rules

Def: a rule of the form A → B is called a unit rule or a chain rule.

- **Note: if A → B then $\alpha A \beta \rightarrow \alpha B \beta$ does not increase the length of the sentential form.**

Problem: Is it possible to avoid unit-rules ?

Ex: A → aA | a | B   B → bB | b | C

=> A → B → bB   A → bB
→ b ==> replace A → B by 3 rules: A → b
→ C   A → C

Problem: A → B removed but new unit rule A → C generated.

Find potential unit-rules.

Def: G: a CFG w/o ε-rules. $A \in N$ (A is a nonterminal).

Define $CH(A) = \{B \in N \mid A \to^* B\}$

- Note: since G contains no ε-rules. $A \to^* B$ iff all rules applied in the derivation are unit-rules.

Problem: how to find CH(A) for all $A \in N$.

Sol: Let $CH_K(A) = \{B \in N \mid \exists n \le k, A \to^n B\}$ Then

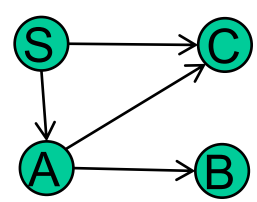
1. $CH_0(A) = \{A\}$ **since $A \to^0 \alpha$ iff $\alpha = A$.**
2. $CH_{k+1}(A) = CH_K(A) \cup \{C \mid B \to C \in P \text{ and } B \in CH_K(A)\}$.
3. $CH(A) = \bigcup_{k \ge 0} CH_k(A)$.

Ex: G: S → ACA |AC|CA|AA|A|C   A → aAa | aa | B | C

B → bB | b   C → cC | c

==> CH(S) = ?   CH(A) = ?

CH(B) = ?   CH(C) = ?



Removing Unit-rules

Theorem 2.3: G: a CFG w/o ε-rules. Then there is a CFG H' equivalent to G but contains no unit-rules.

Pf: H'' and H' are constructed as follows:

1. Let P'' = P U { A → w | B ∈ CH(A) and B → w ∈ P }. and
2. let P' = P'' with all unit-rules removed.

By lem 1.4, L(H'') = L(G). the proof that L(H'') = L(H') is similar to Theorem 1.5. left as an exercise (Hint: Unit rules applied in a derivation can always be decreased to zero**)**.

Ex: G: S → ACA |AC|CA|AA|A|C A → aAa | aa | B | C

B → bB | b C → cC | c

==>CH(S)={S,A,C,B}, CH(A) = {A,B,C}, CH(B) ={B}, CH(C) = {C}.

Hence P'' = P U { …. ? } and

P' = { ? }.

Note: if G contains no ε-rules, then so does H'.

Contracting Grammars

- **Given a CFG G , it would be better to replace G by another G' if G' contains fewer nonterminal symbols and/or production rules.**
  - **Like FAs, where inaccessible states can be removed, some symbols and rules in a CFG can be removed w/t affecting its accepted language.**

Def: A nonterminal A in a CFG G is said to be grounding if it can derive terminal strings. (i.e., there is w $\in \Sigma^*$ s.t. $A \rightarrow^* w$.} O/W we say A is nongrounding.

Note: Nongrounding symbols (and all rules using nonground symbols) can be removed from the grammars.

Ex: G: S → a | aS | bB   B → C | D | aB | BC

==> Only S is grounding and B,C, D are nongrounding

==> B,C,D and related rules can be removed from G.

==> G can be reduced to: S → a | aS

Finding nongrounding symbols

Given a CFG G = (N,S,P,S). the set of grounding symbols can be defined inductively as follows:

1. Init: If there is a rule A→w in P s.t. w ∈ Σ*, then A is grounding.

2. ind.: If A → w is a rule in P s.t. each symbol in w is either a terminal or grounding then A is grounding.

Exercise: According to the above definition, write an algorithm to find all grounding (and nongrounding) symbols for arbitrarily given CFG.

Ex: S → aS | b |cA | B | C | D A → aC | cD | Dc | bBB

B → cC | D |b C → cC | D D → cD | dC

=> By init: S, B is grounding => S,B,A is grounding

=> G can be reduced to :

S → aS | b |cA | B A → bBB B → b

Unreachable symbols

Def: a nonterminal symbol A in a CFG G is said to be reachable iff it occurs in some sentential form of G. i.e., there are α,β s.t. $S \rightarrow^* \alpha A\beta$. It A is not reachable, it is said to be unreachable.

- Note: Both nongrounding symbols and unreachable symbol are useless in the sense that they can be removed from the grammars w/o affecting the language accepted.

Problem: How to find reachable symbols in a CFG ?

Sol: The set of all reachable symbols in G is the least subset R of N s.t. 1. the start symbol $S \in R$, and

2. if $A \in R$ and $A \rightarrow \alpha B\beta \in P$, then $B \in R$.

Ex: S → AC |BS | B A → aA|aF B → |CF | b C → cC | D

D → aD | BD | C E → aA |BSA F → bB |b.

=> R = {S, A,B,C,F,D} and E is unreachable.

Elimination of empty and unit productions

- The removal of ε-rules and unit-rules can be done simultaneously.
- **G = (N,S,P,S) : a CFG. The EU-closure of P, denoted EU(P), is the least set of rules including P s.t.**
  1. **If $A \rightarrow \alpha B \beta$ and $B \rightarrow \varepsilon \in$ EU(P) then $A \rightarrow \alpha\beta \in$ EU(P).**
  2. **If $A \rightarrow B \in$ EU(P) and $B \rightarrow \gamma \in$ EU(P) then $A \rightarrow \gamma \in$ EU(P).**
  - Quiz: What is the recursive definition of EU(P) ?
- **Notes:**
  - 1. EU(P) exists and is finite.
  - If $A \rightarrow \alpha_0 A_1 \alpha_1 A_2 \ldots A_n \alpha_n$ contains n nonterminals on the RHS ==> there are at most 2^n-1 new rules which can be added to EU(P), due to (a) and this rule.
  - If $B \rightarrow \gamma \in$ P and |N| = n then there are at most n-1 rules can be added to EU(P) due to this rule and (b).
  - 2. It is easy to find EU(P).

EU-closure of production rules

- Procedure EU(P)

1. P' = P; NP = {};
2. for each ε-rule B → ε ∈ P' do
   for each rule A → αBβ do
   NP = NP U {A → αβ };
3. for each unit rule A → B ∈ P' where B ≠ A,
   for each rule B → γ do
   NP = NP U {A → γ};
4. If NP ⊆ P' then return (P')
   else{P' = P' U NP; NP = {};
   goto 2}

Notation: let $P'_k =_{def}$ the value of P' after the kth iteration of statement 2 and 3.

Ex 21.5': P={ S → [S] | SS | ε }

1+3 => S → [] --- 4.

2+3 => S → S, S → S --- 5.

=> EU(P) = P U { S → [], S → S }

Equivalence of P and EU(P) (skipped!).

- $G = (N,\Sigma,P,S)$, $G' = (N,\Sigma,EU(P), S)$.

Lem 1: for each rule $A \rightarrow \gamma \in EU(P)$, we have $A \rightarrow^*_G \gamma$.

pf: By ind on k where k is the number of iteration of statement 2,3 of the program at which $A \rightarrow \gamma$ is obtained.

1. $k = 0$. then $A \rightarrow \gamma \in EU(P)$ iff $A \rightarrow \gamma \in P$. Hence $A \rightarrow^*_G \gamma$.
2. $K = n+1 > 0$.

2.1: $A \rightarrow \gamma$ is obtained from statement 2.

==> $\exists$ B, α, β with $\alpha\beta = \gamma$ s.t. $A \rightarrow \alpha B \beta$ and $B \rightarrow \varepsilon \in P'_n$.

□ Hence $A \rightarrow^*_G \alpha B \beta \rightarrow^*_G \alpha\beta = \gamma$.

2.2 $A \rightarrow \gamma$ is obtained from statement 3.

==> $\exists$ $A \rightarrow B$ and $B \rightarrow \gamma \in P'_n$.

□ Hence $A \rightarrow^*_G B \rightarrow^*_G \gamma$.

Corollary: $L(G) = L(G')$.

S can never occur at RHS (skipped!!)

- G = (N,Σ,P,S) : a CFG. Then there exists a CFG G' = (N',Σ,P',S') s.t. (1) L(G') = L(G) and (2) the start symbol S' of G' does not occur at the RHS of all rules of P'.

Ex: G: S → aS | AB |AC　　　A → aA | ε
　　　B → bB | bS　　　C → cC | ε.

==> G': S' → aS | AB |AC
　　　S → aS | AB |AC　　　A → aA | ε
　　　B → bB | bS　　　C → cC | ε.

ie., Let G' = G if S does not occurs at the RHD of rules of G.

o/w: let N' = N U {S'} where S' is a new nonterminal ∉ N.

and Let P' = P U {S' → α | S → α ∈ P }.

It is easy to see that G' satisfies condition (2). Moreover for any α∈(N UΣ)*, we have $S' \rightarrow^{+}_{G'} \alpha$ iff $S \rightarrow^{+}_{G} \alpha$.

Hence L(G) = L(G').

Generality of Greibach normal form (skipped!)

- The topic about Greibach normal form will be skipped!
  - Content reserved for self study.
- Claim: Every CFG G can be transformed into an equivalent one G' in gnf form (i.e., L(G') = L(G) - { ε }).

Definition: (left-most derivation)

- $\alpha,\beta \in (N \cup \Sigma)^*$: two sentential forms
- $\alpha \ {}^{L}\!\!-\!\!-\!\!>_G \beta \ =_{def} \ \exists x \in \Sigma^*, A \in N, \gamma \in (N\cup\Sigma)^*$, rule $A \to \delta$ s.t.
- $\alpha = x\,A\,\gamma$ and $\beta = x\,\delta\,\gamma$.
- i.e., $\alpha \ {}^{L}\!\!-\!\!-\!\!> \beta$ iff $\alpha \dashrightarrow \beta$ and the left-most nonterminal symbol A of β is replaced by the rhs δ of some rule A-> δ.

- Derivations and left-most derivations:
  - Note: $^{L}\!\!-\!\!-\!\!>_G \subseteq -\!\!-\!\!>_G$ but not the converse in general !
  - Ex: G : A -> Ba | ABc; B -> a | Ab
  - then aAb B --> aAb Ba and aAbB --> a Ba bB and
  - aAbB $^{L}\!\!-\!\!-\!\!>$ a Ba bB but not aAb B $^{L}\!\!-\!\!-\!\!>$ aAb Ba

Left-most derivations

- As usual, let $\overset{L}{-->}^*_G$ be the ref. and trans. closure of $\overset{L}{-->}_G$.

- Equivalence of derivations and left-most derivations :

Theorem: A: a nonterminal; x: a terminal string. Then

$$A \dashrightarrow^* x \text{ iff } A \overset{L}{\dashrightarrow}^* x.$$

pf: (<=:) trivial. Since $\overset{L}{-->} \subseteq -->$ implies $\overset{L}{-->}^* \subseteq -->^*$.

(=>:) left as an exercise.

(It is easier to prove using parse tree.)

Transform CFG to gnf

- G= (N,Σ,P,S) : a CFG where each rule has the form:
  - A -> a or
  - A -> B_1 B_2 …B_n (n > 1). // we can transform every cfg into such from if it has no ε-rule.
- Now for each pair (A, a) with A ∈ N and a ∈ Σ, define the set R(A,a) $=_{def}$ { β ∈ N* | A ^{L}->* a β }.

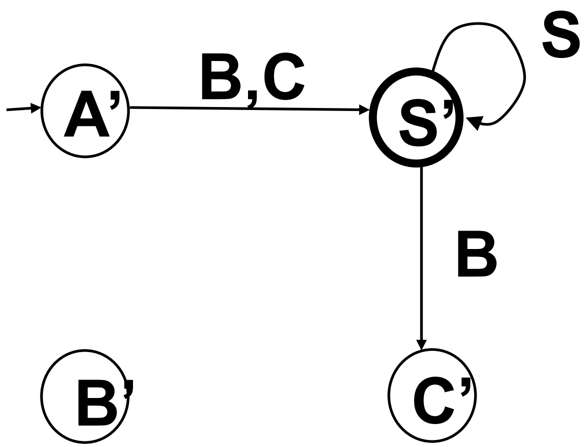
Ex: If G1 = { S-> AB | AC | SS, C-> SB, A->[, B ->] }, then
  - CSSB ∈ R(C,[) since
  - C ^{L}-->SB ^{L}--> SS B ^{L}-->SS SB ^{L}--> ACSSB ^{L}--> [CSSB
- Claim: The set R(A,a) is regular over N*. In fact it can be generated by the following left-linear grammar:
- G(A,a) = (N', Σ',P',S') where
  - N' = {X' | X ∈ N}, Σ' = N, S' = A' is the new start symbol,
  - P' = { X' -> Y'ω | X -> Yω ∈ P } U { X' -> ε | X -> a ∈ P }

- Ex: For G1, the CFG G1(C, [ ) has
  - nonterminals: S', A',B',C',
  - terminals: S,A,B,C,
  - start symbol: C'
  - rules P' = { S'-> A'B | A'C | S'S, C'-> S'B, A'->ε }
  - cf: P = { S -> AB | AC | SS, C-> SB, A->[, B ->] }
- Note: Since G(A,a) is regular, there is a strongly right linear grammar equivalent to it. Let G'(A,a) be one of such grammar. Note every rule in G'(A,a) has the form X' -> BY' or X' ->ε }
- let $S_{(A,a)}$ be the start symbol of the grammar G'(A,a).
- let G_1 = G U $U_{A \in N, a \in \Sigma}$ G'(A,a) with terminal set Σ,
  - and nonterminal set: N U nonterminals of all G'(A,a).
  - 1. Rules in G_1 have the forms: X -> b, X->Bω or X -> e
  - 2. L(G) = L(G_1) since no new nonterminals can be derived from S, the start symbol of G and G_1.

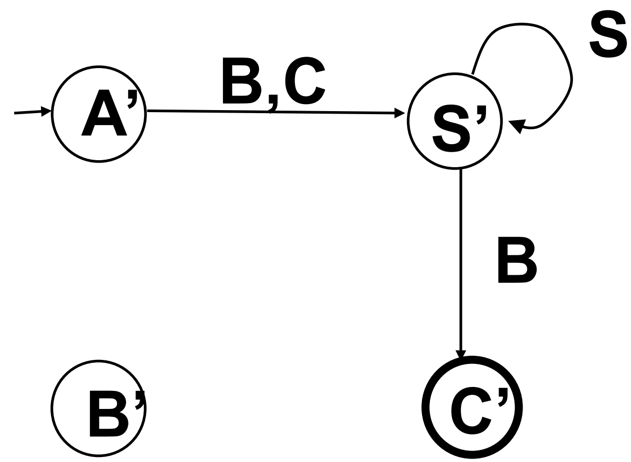
Example:

- From G1, we have:
  - R(S, [) = ?   R(C,[) = ?   R(A,[) = ?   R(B,[) = ?
  - All four grammar G(S,[), G(A,[), G(A, [) and G(B,[) have the same rules:
  - { S' -> A'B | A'C | S'S,   C' -> S'B,   A' -> ε }, but
  - with different start symbols: S', C', A' and B'.
  - The FAs corresponding to All G(A,a) have the same transitions and common initial state (A').
  - They differs only on the final state.
- Exercises:
  1. Find the common grammar rules corresponding to G(S,]), G(C,]), G(A,]) and G(B,])
  2. Draw All FAs corresponding to R(S,]), R(C,]), R(A,]) and R(B,]), respectively.
  3. Find regular expressions equivalent to the above four sets.

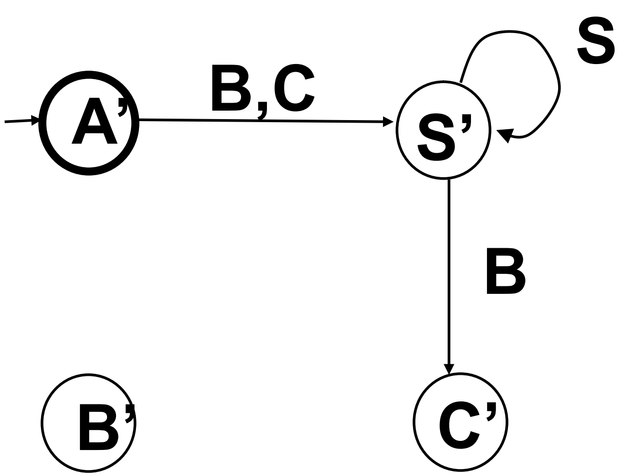
FAs corresponding to various G(A,[)s.



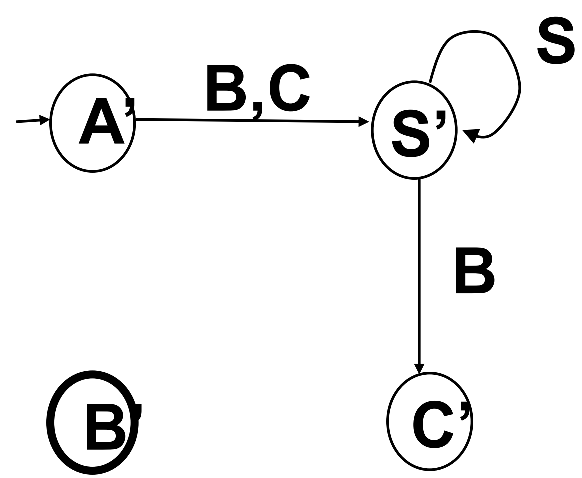
R(S,[) = (B+C)S*



R(C,[) = (B+C)S* B

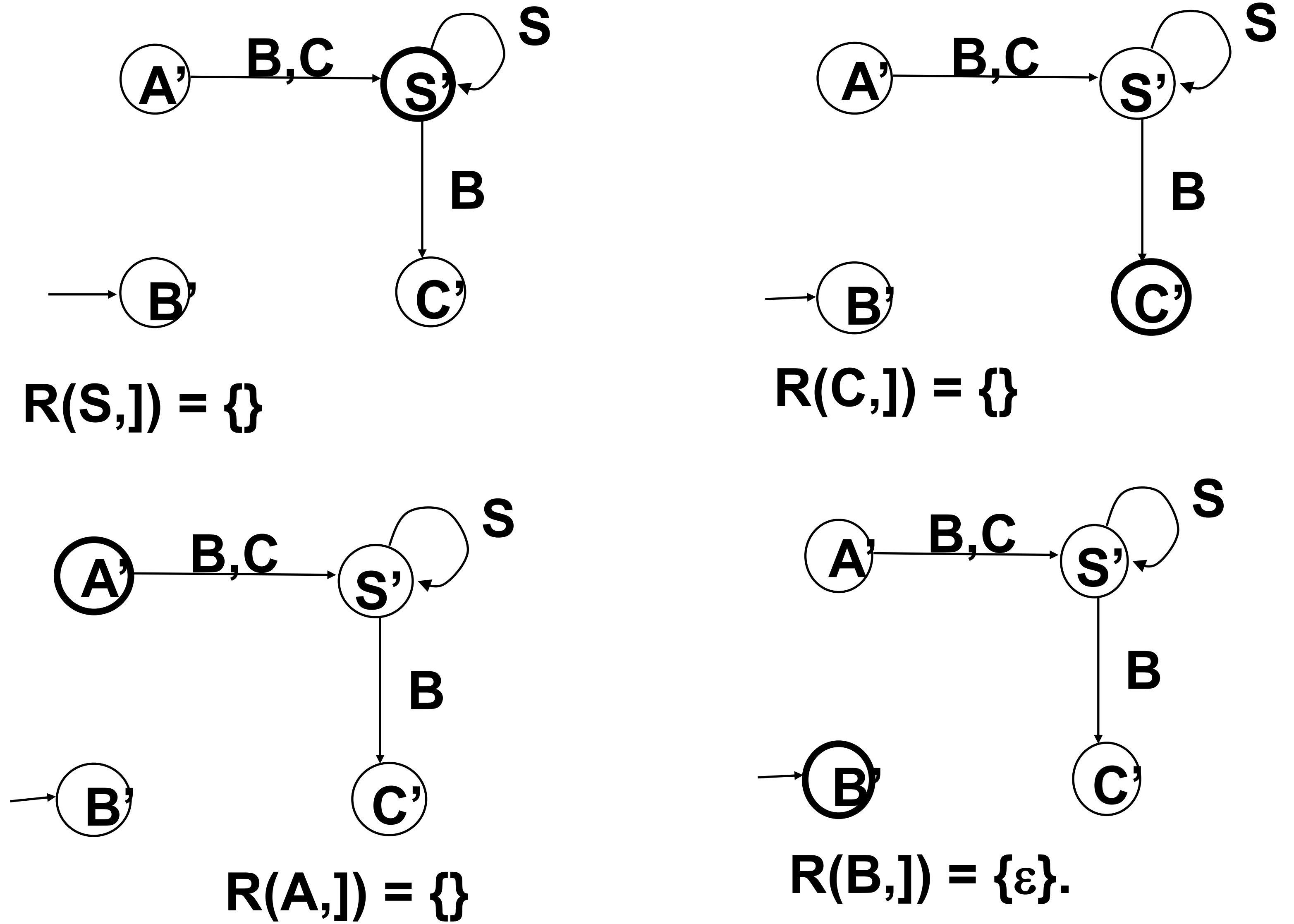


R(A,[) = {ε}



R(B,[) = {}.

FAs corresponding to various G(A,])s.



common rules: S' -> A'B | A'C | S'S, C' -> S'B, B' -> ε

Strongly right linear grammar corresponding to G(A,a)s

- $G'(S,[) = \{ S_{(S,[)} \to BX \mid CX \quad X \to SX \mid \varepsilon \}$
- $G'(C,[) = \{ S_{(C,[)} \to BY \mid CY \quad Y \to SY \mid BZ,\ Z \to \varepsilon \}$
- $G'(A,[) = \{ S_{(A,[)} \to \varepsilon \}$
- $G'(B,[) = G'(S,]) = G'(C,]) = G'(A,]) = \{\}$
- $G'(B,]) = \{ S_{(B,])} \to \varepsilon \}$

- Let $G_2 = G_1$ with every rule of the form:

  $X \to B\omega$

  replaced by the productions $X \to b\ S_{(B,b)}\omega$ for all b in Σ.
- Note: every production of G2 has the form:

  $X \to b$ or $X \to \varepsilon$ or $X \to b\ S_{(B,b)}\ \omega$.

Let G_3 = the resulting CFG by applying ε rule-elimination to G_2.

Now it is easy to see that $L(G) = L(G_1)$ =?= $L(G_2) = L(G_3)$.

and G3 is in gnf.

From G_1 to G_2

By def. $G1_1 = G1 \cup U_{X \text{ in } N, a \text{ in } \Sigma}\ G1(X, a)$

= G1 U { $S_{(S,[)}$ -> BX | CX   X -> SX | ε } U

 { $S_{(C,[)}$ -> BY | CY   Y -> SY | BZ, Z -> ε } U

 { $S_{(A,[)}$ -> ε } U

 { $S_{(B,])}$ -> ε }

 Note: $L(G1_1) = L(G1)$ why ?

and $G1_2$ = { S -> [$S_{(A,[)}$ B |] $S_{(A,])}$ B | // S -> AB

[$S_{(A,[)}$ C |] $S_{(A,])}$ C | // S-> AC

[$S_{(S,[)}$ S |] $S_{(S,])}$ S   // S-> SS,

C-> [$S_{(S,[)}$ B |] $S_{(S,])}$ B   // C-> SB,

A->[, B ->] } U ….

/* { $S_{(S,[)}$ -> BX | CX   X -> SX ε } U

* { $S_{(C,[)}$ -> BY | CY   Y -> SY | BZ, Z -> ε } U

*/ { $S_{(A,[)}$ -> ε } U { $S_{(B,])}$ -> ε }

From G_2 to G_3

- By applying ε-rule elimination to $G1_2$, we can get $G1_3$:
- First determine all nullable symbols: X, Z, $S_{(A,[)}$, $S_{(B,])}$

$G1_2$ = { S -> [$S_{(A,[)}$ B | [$S_{(A,[)}$ C | [$S_{(S,[)}$ S

C-> [$S_{(S,[)}$ B

A-> [, B ->] } U

{ $S_{(S,[)}$ ->] $S_{(B,])}$ X | [$S_{(C,[)}$ X // BX| CX

X -> [$S_{(S,[)}$ X | ε } U

{ $S_{(C,[)}$ ->] $S_{(B,])}$ Y | [$S_{(C,[)}$ Y // BY | CY

Y -> [$S_{(S,[)}$ Y |] $S_{(B,])}$ Z // SY | BZ,

Z -> ε } U { $S_{(A,[)}$ -> ε , $S_{(B,])}$ -> ε }

Hence $G1_3$ = ?

$G1_3$

$G1_3$ = { S -> [B | [C | [$S_{(S,[)}$ S

C-> [$S_{(S,[)}$ B

A-> [, B ->]

$S_{(S,[)}$ ->] X |] | [$S_{(C,[)}$ X | [$S_{(C,[)}$ // BX| CX

X -> [$S_{(S,[)}$ X | [$S_{(S,[)}$ X } U

$S_{(C,[)}$ ->] Y | [$S_{(C,[)}$ Y

Y ->]$S_{(B,])}$ Y |] } //SY | BZ,

Lemma 21.7: For any nonterminal X and x in Σ^*,

X $^{L}\text{-->}^*_{G1}$ x iff X $^{L}\text{-->}^*_{G2}$ x.

Pf: by induction on n s.t. X ->^n_{G1} x.

Case 1: n = 1. then the rule applied must be of the form:

X -> b or X -> ε.

But these rules are the same in both grammars.

Equivalence of G_1 and G_2

- **Inductive case: n > 1.**

  $X \xrightarrow{L}_{G1} B\omega \xrightarrow{L}^{*}_{G1} by = x$ iff

  $X \xrightarrow{L}_{G1} B\omega \xrightarrow{L}^{*}_{G1} bB_1B_2\ldots B_k\,\omega \xrightarrow{L}^{*}_{G1} bz_1\ldots z_k\,z = x$, where

   $bB_1B_2\ldots B_k\,\omega$ is the first sentential form in the sequence in which b appears and $B_1B_2\ldots B_k$ belongs to R(B,b),

iff (by definition of R(B,b) and G(B,b))

$X \xrightarrow{L}_{G2} b\,S_{(B,b)}\,\omega \xrightarrow{L}^{*}_{G1} b\,B_1B_2\ldots B_k\,\omega \xrightarrow{L}^{*}_{G1} bz_1\ldots z_k\,z$, where the subderivation $S_{(B,b)} \xrightarrow{L}^{*}_{G1} B_1B_2\ldots B_k$ is a derivation in $G(B,b) \subseteq G1 \cap G2$.

iff $X \xrightarrow{L}^{*}_{G2} b\,S_{(B,b)}\,\omega \xrightarrow{L}^{*}_{G2} b\,B_1B_2\ldots B_k\,\omega \xrightarrow{L}^{*}_{G1} bz_1\ldots z_k\,z = x$

But by ind. hyp., $B_j \xrightarrow{L}^{*}_{G2} z_j$ ($0 < j < k+1$) and $\omega \xrightarrow{L}^{*}_{G2} y$.

Hence $X \xrightarrow{L}^{*}_{G2} x$.