Section: Transforming grammars (Ch. 6)

Methods for Transforming Grammars

We will consider CFL without λ. It would be easy to add λ to any grammar by adding a new start symbol $\mathbf{S}_0$,

$$\mathbf{S}_0 \rightarrow \mathbf{S} \mid \lambda$$

Theorem (Substitution) Let G be a CFG. Suppose G contains

$$\mathbf{A} \rightarrow x_1 \mathbf{B} x_2$$

where A and B are different variables, and B has the productions

$$\mathbf{B} \rightarrow y_1 | y_2 | \ldots | y_n$$

Then can construct G' from G by deleting

$$\mathbf{A} \rightarrow x_1 \mathbf{B} x_2$$

from P and adding to it

$$\mathbf{A} \rightarrow x_1 y_1 x_2 | x_1 y_2 x_2 | \ldots | x_1 y_n x_2$$

Then, L(G)=L(G').

Example:

S → aBa
B → aS | a

becomes

$S \to aaSa \mid aaa$
$B \to aS \mid a$

useless rules

Definition: A production of the form $A \to Ax$, $A \in V$, $x \in (V \cup T)^*$ is *left recursive*.

Example Previous expression grammar was left recursive.

$$\mathbf{E} \rightarrow \mathbf{E}+\mathbf{T} \mid \mathbf{T}$$
$$\mathbf{T} \rightarrow \mathbf{T}*\mathbf{F} \mid \mathbf{F}$$
$$\mathbf{F} \rightarrow \mathbf{I} \mid (\mathbf{E})$$
$$\mathbf{I} \rightarrow \mathbf{a} \mid \mathbf{b}$$

Derivation of a+b+a+a is:

$$\mathbf{E} \Rightarrow \mathbf{E}+\mathbf{T} \Rightarrow \mathbf{E}+\mathbf{T}+\mathbf{T} \Rightarrow \mathbf{E}+\mathbf{T}+\mathbf{T}+\mathbf{T} \overset{*}{\Rightarrow} \mathbf{a}+\mathbf{T}+\mathbf{T}+\mathbf{T}$$

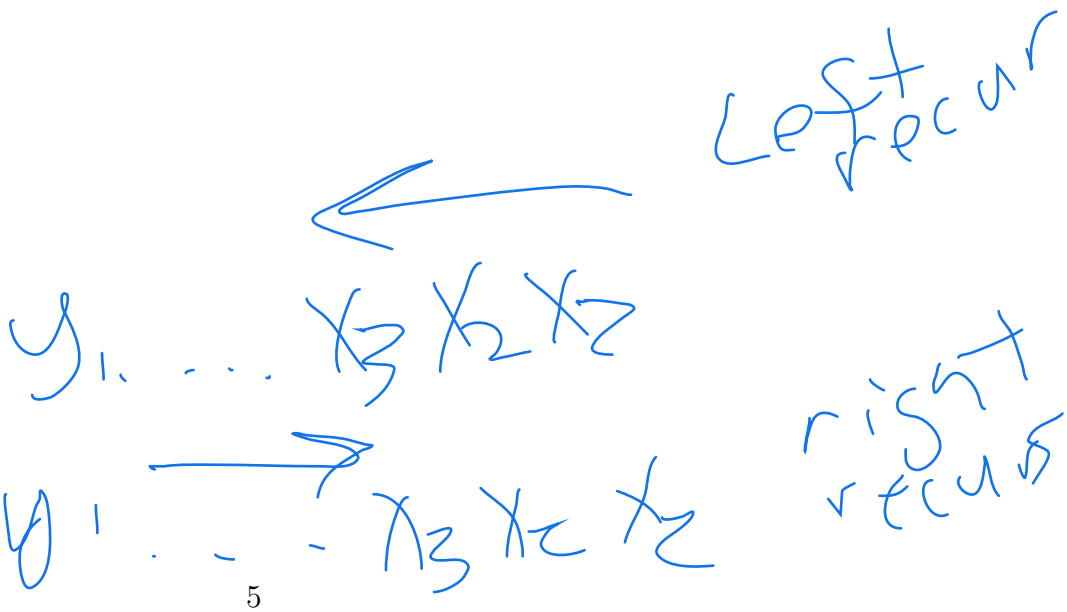
Theorem (Removing Left recursion) Let G=(V,T,S,P) be a CFG. Divide productions for variable A into left-recursive and non left-recursive productions:

$$\mathbf{A} \to \mathbf{A}x_1 \mid \mathbf{A}x_2 \mid \ldots \mid \mathbf{A}x_n$$
$$\mathbf{A} \to y_1 | y_2 | \ldots | y_m$$

where x_i, y_i are in $(\mathbf{V} \cup \mathbf{T})^*$.

Then G'=(V$\cup${Z}, T, S, P') and P' replaces rules of form above by

$$\mathbf{A} \to y_i | y_i\mathbf{Z}, \ \mathbf{i=1,2,\ldots,m}$$
$$\mathbf{Z} \to x_i | x_i\mathbf{Z}, \ \mathbf{i=1,2,\ldots,n}$$



Example:

$\mathbf{E \rightarrow E{+}T|T}$ **becomes** $E \rightarrow TZ \mid T$, $Z \rightarrow +TZ \mid +T$

$\mathbf{T \rightarrow T{*}F|F}$ **becomes** $T \rightarrow FY \mid F$, $Y \rightarrow *FY \mid *F$

other rule

$F \rightarrow I$

$I \rightarrow a \mid b$

Now, Derivation of a+b+a+a is:

$E \rightarrow TZ \rightarrow FZ \rightarrow IZ \rightarrow aZ \rightarrow a+TZ \ldots$ we didn't see any other terminal

Useless productions

$S \rightarrow aB \mid bA$
$A \rightarrow aA$
$B \rightarrow Sa$
$C \rightarrow cBc \mid a$

everything useless

never get to C

What can you say about this grammar?

useless variables
useless rules

Theorem (useless productions) Let G be a CFG. Then ∃ G' that does not contain any useless variables or productions s.t. L(G)=L(G').

To Remove Useless Productions:

Let G=(V,T,S,P).

I. Compute V_1={Variables that can derive strings of terminals}

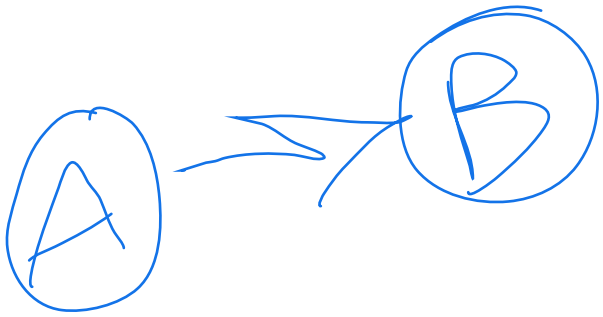
1. **$V_1=\emptyset$**
2. **Repeat until no more variables added**
   - **For every A$\in$V with A$\rightarrow x_1x_2\ldots x_n$, $x_i \in$($T^* \cup V_1$), add A to V_1**
3. **P_1 = all productions in P with symbols in $(V_1 \cup T)^*$**

Then G_1=(V_1,T,S,P_1) has no variables that can't derive strings.

II. Draw Variable Dependency Graph

For A $\rightarrow$ xBy, draw A$\rightarrow$B.

Remove productions for V if there is no path from S to V in the dependency graph. Resulting Grammar G' is s.t. L(G)=L(G') and G' has no useless productions.



Example:

$\mathbf{S \to aB \mid \cancel{bA}}$

$\mathbf{\cancel{A \to aA}}$

$\mathbf{B \to Sa \mid b}$

$\mathbf{C \to cBc \mid a}$

$\mathbf{D \to bCb}$

$\mathbf{E \to \cancel{Aa} \mid b}$

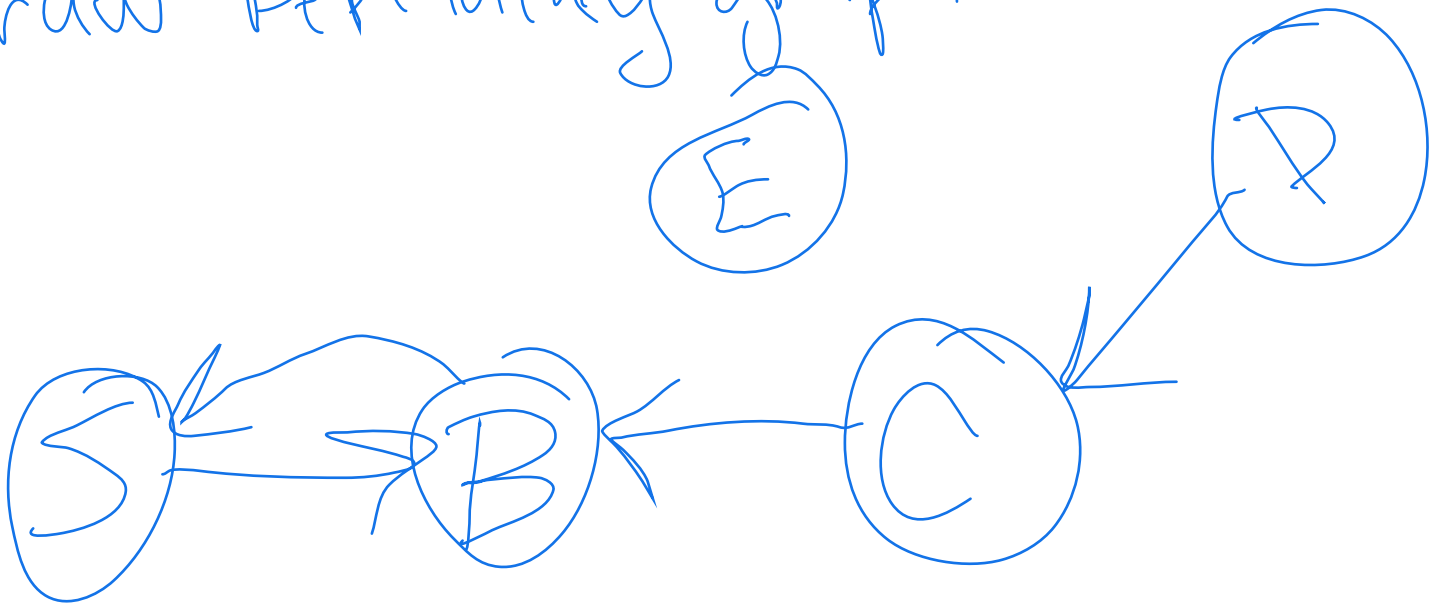
$S \to aB$

$B \to Sa \mid b$

not A

$V_1 = \{B, C, E, D, S\}$

Draw Pependency graph



From S, can reach B

Theorem (remove λ productions) Let G be a CFG with λ not in L(G). Then $\exists$ a CFG G' having no λ-productions s.t. L(G)=L(G').

To Remove λ-productions

1. **Let $V_n = \{$A $\mid \exists$ production A$\rightarrow\lambda$ $\}$**
2. **Repeat until no more additions**
   - **if B$\rightarrow$A$_1$A$_2$...A$_m$ and A$_i\in V_n$ for all i, then put B in V_n**
3. **Construct G' with productions P' s.t.**
   - **If A$\rightarrow x_1x_2\ldots x_m \in$P, $m \geq 1$, then put all productions formed when x_j is replaced by λ (for all $x_j \in V_n$) s.t. $|$rhs$| \geq 1$ into P'.**

Example:

$\mathbf{S} \rightarrow \mathbf{Ab}$
$\mathbf{A} \rightarrow \mathbf{BCB} \mid \mathbf{Aa}$
$\mathbf{B} \rightarrow \mathbf{b} \mid \lambda$
$\mathbf{C} \rightarrow \mathbf{cC} \mid \lambda$

$V_n = \{B, C, A\}$

$S \rightarrow Ab \mid b$

$A \rightarrow BCB \mid BC \mid CB \mid BB \mid B \mid C$

$A \rightarrow Aa \mid a$

$B \rightarrow b$

$C \rightarrow cC \mid c$

Definition Unit Production

$$A \rightarrow B$$

where A,B $\in$V.

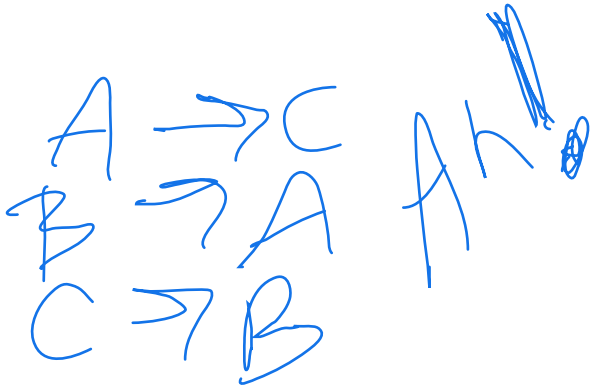
Consider removing unit productions:

Suppose we have

A $\rightarrow$ B **becomes**
B $\rightarrow$ a | ab

But what if we have

A $\rightarrow$ B **becomes**
B $\rightarrow$ C
C $\rightarrow$ A



Stopped here

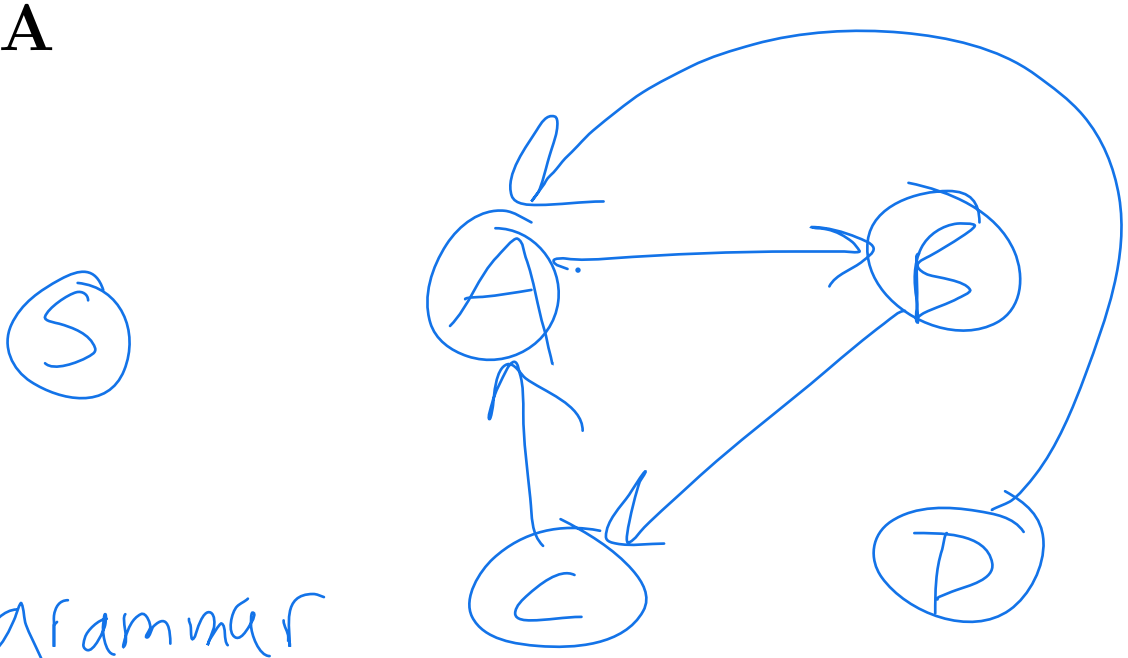
Theorem (Remove unit productions) Let G=(V,T,S,P) be a CFG without λ-productions. Then $\exists$ CFG G'=(V',T',S,P') that does not have any unit-productions and L(G)=L(G').

To Remove Unit Productions:

1. **Find for each A, all B s.t. A $\stackrel{*}{\Rightarrow}$B (Draw a dependency graph)**
2. **Construct G'=(V',T',S,P') by**
   (a) **Put all non-unit productions in P'**
   (b) **For all A$\stackrel{*}{\Rightarrow}$B s.t. B$\rightarrow y_1|y_2|\ldots y_n \in$ P', put A$\rightarrow y_1|y_2|\ldots y_n \in$ P'**

Example:

$\mathbf{S \rightarrow AB}$
$\mathbf{A \rightarrow B}$
$\mathbf{B \rightarrow C \mid Bb}$
$\mathbf{C \rightarrow A \mid c \mid Da}$
$\mathbf{D \rightarrow A}$



New grammar

From 2a $S \rightarrow AB$
$B \rightarrow Bb$
$C \rightarrow c \mid Da$

From 2b $A \rightarrow c \mid Bb \mid Da$
$B \rightarrow c \mid Da$
$C \rightarrow Bb$
$D \rightarrow c \mid Da \mid Bb$

Theorem Let L be a CFL that does not contain λ. Then $\exists$ a CFG for L that does not have any useless productions, λ-productions, or unit-productions.

Proof

1. **Remove λ-productions**
2. **Remove unit-productions**
3. **Remove useless productions**

Note order is very important. Removing λ-productions can create unit-productions! QED.

great for Brute Force Parsers

Definition: A CFG is in Chomsky Normal Form (CNF) if all productions are of the form

$$\mathbf{A \to BC \quad or \ A \to a}$$

where A,B,C$\in$V and a$\in$T.

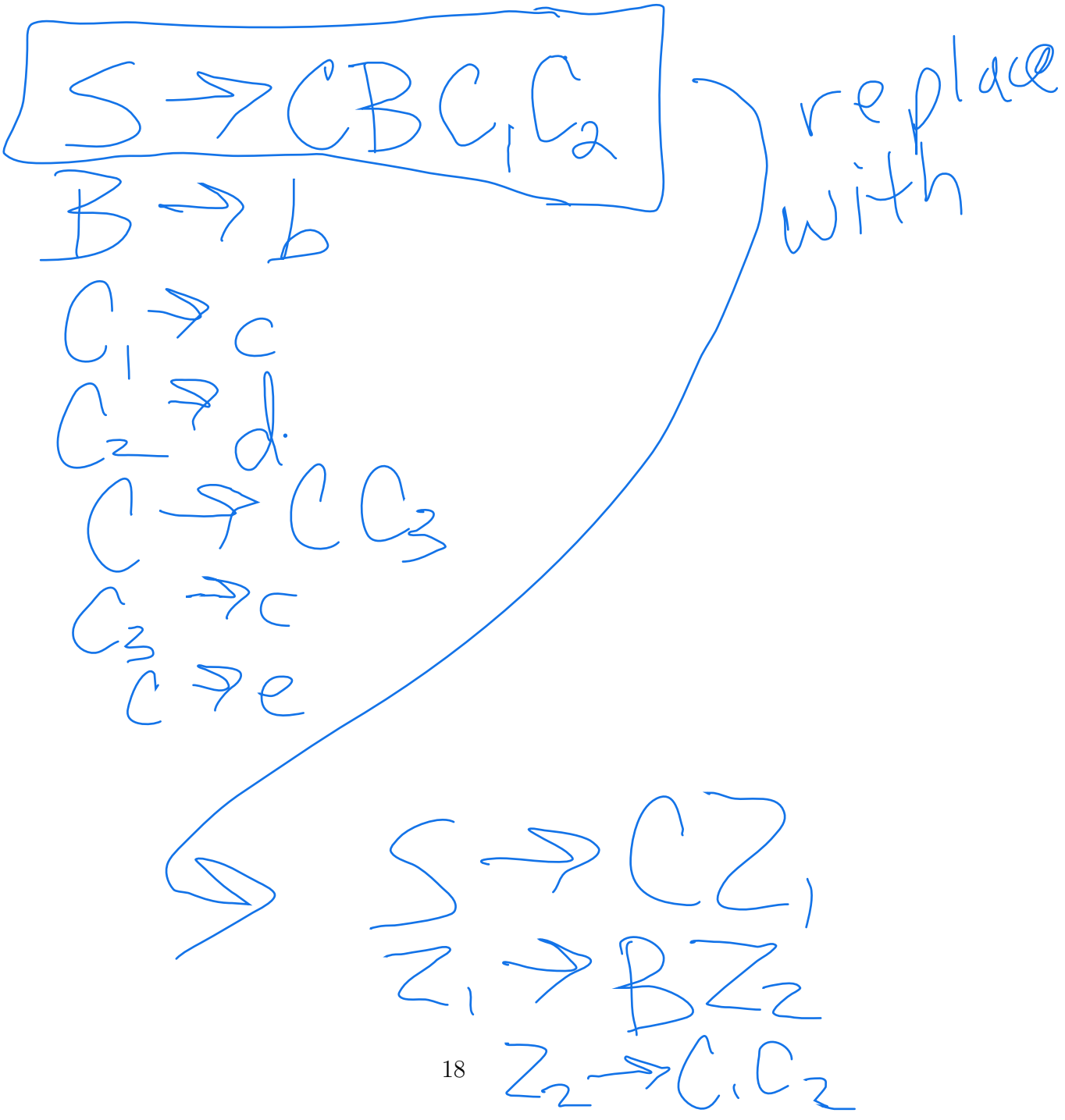
Theorem: Any CFG G with λ not in L(G) has an equivalent grammar in CNF.

Proof:

1. **Remove λ-productions, unit productions, and useless productions.**
2. **For every rhs of length > 1, replace each terminal x_i by a new variable C_j and add the production $C_j \to x_i$.**
3. **Replace every rhs of length > 2 by a series of productions, each with rhs of length 2. QED.**

Example:

$$\mathbf{S \rightarrow CBcd}$$
$$\mathbf{B \rightarrow b}$$
$$\mathbf{C \rightarrow Cc \mid e}$$



Definition: A CFG is in Greibach normal form (GNF) if all productions have the form

$$\mathbf{A} \rightarrow \mathbf{ax}$$

where a$\in$T and x$\in V^*$

Theorem For every CFG G with λ not in L(G), $\exists$ a grammar in GNF.

Proof:

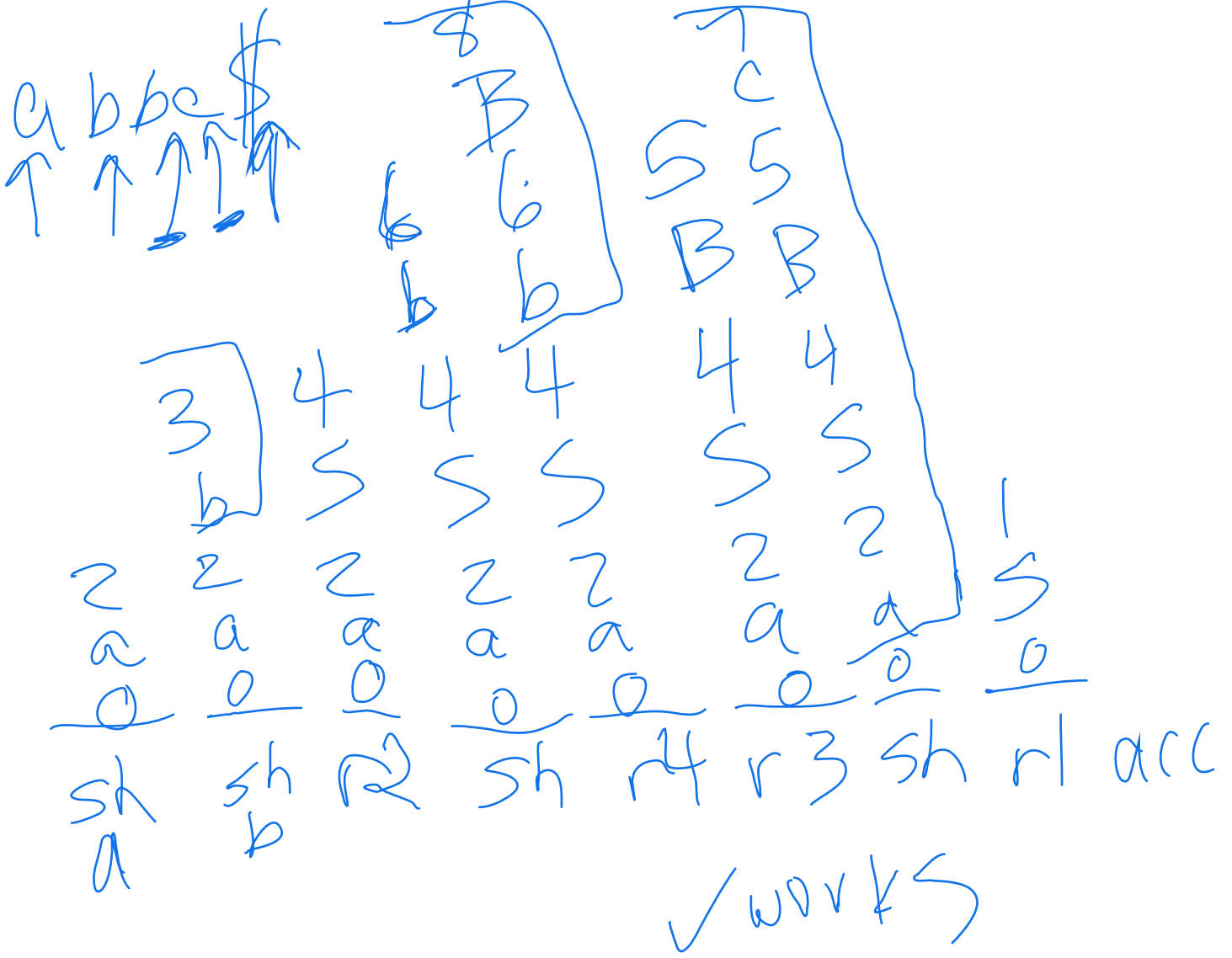
1. **Rewrite grammar in CNF.**
2. **Relabel Variables A_1, A_2, ... A_n**

3. **Eliminate left recursion and use substitution to get all productions into the form:**

$$A_i \rightarrow A_j x_j, \; j > i$$
$$Z_i \rightarrow A_j x_j, \; j \leq n$$
$$A_i \rightarrow \mathbf{a} x_i$$

   where a$\in$T, $x_i \in$V*, and Z_i are new variables introduced for left recursion.

4. **All productions with A_n are in the correct form, $A_n \rightarrow \mathbf{a} x_n$. Use these productions as substitutions to get A_{n-1} productions in the correct form. Repeat with A_{n-2}, A_{n-3}, etc until all productions are in the correct form.**


a b b c $
S B C
S B B
3 4 4 4 4 4
b S S S S S
z z z z z z z
a a a a a a a
0 0 0 0 0 0 0 0
sh sh r2 sh r4 r3 sh r1 acc
a b
works