Normal Form Grammar

Normal Forms facilitate the analysis of grammars and languages

Normal Forms impose restrictions on the ***CFG***

Normal Forms generates the entire ***CFG***

Algorithms transform ***CFG*** into an equivalent grammar in ***Normal Form*** via ***a sequence*** of rule additions, deletions, or modifications

Problems:

Eliminate Recursive Start Symbol From Grammar

Recursive start symbol S defined as $S \overset{+}{\Rightarrow} uSv$ enables sentential symbols in intermediate results of a derivation

Lemma 4.1.1

For $G = (V, \Sigma, P, S)$ a CFG, a G' that satisfies $L(G) = L(G')$ and the start symbol S' in G' is not recursive

$G' = (V', \Sigma, P', S')$ where $V' = V \cup \{S'\}$ and $P' = P \cup \{S' \rightarrow S\}$

$\boldsymbol{S} \overset{*}{\Rightarrow} \boldsymbol{u}$ in $\boldsymbol{G}$ and $\boldsymbol{S}' \overset{*}{\Rightarrow} \boldsymbol{S} \Rightarrow \boldsymbol{u}$ in $\boldsymbol{G'}$

Problems:

Eliminate *Recursive Start Symbol* From Grammar

Example 4.1.1

$G = (V, \Sigma, P, S)$, $V = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$

P: $S \rightarrow aS \mid AB \mid AC$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bB \mid bS$

$C \rightarrow cC \mid \lambda$

$G' = (V', \Sigma, P', S')$, $V' = \{S', S, A, B, C\}$, $\Sigma = \{a, b, c\}$

P': $S' \rightarrow S$

$S \rightarrow aS \mid AB \mid AC$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bB \mid bS$

$C \rightarrow cC \mid \lambda$

Problems:

Adding Rules to Grammars

Lemma 4.1.2

For CFG $G = (V, \Sigma, P, S)$

augment rules

if $A \overset{*}{\Rightarrow} w$ is in G

then $G' = (V, \Sigma, P', S)$ where $P' = P \cup \{A \overset{*}{\Rightarrow} w\}$ and $L(G) \subseteq L(G')$

Lemma 4.1.3

For CFG $G = (V, \Sigma, P, S)$

if $A \overset{*}{\Rightarrow} uBv$ and $B \rightarrow w_1 \mid w_2 \mid \ldots \mid w_n$ are **rules in P**

replace rule

then G' = (V, Σ, P', S) where

$$P' = \{P - \{A \overset{*}{\Rightarrow} uBv\}) \cup \{A \rightarrow u\,w_1\,v \mid u\,w_2\,v \mid \ldots \mid u\,w_n\,v\}$$

Replace $A \overset{*}{\Rightarrow} uBv$ with $A \rightarrow u\,w_1\,v \mid u\,w_2\,v \mid \ldots \mid u\,w_n\,v$

Addition of Rules to Grammars

Lemma 4.1.3 continued

From Lemma 4.12,

A terminal string derivable in G using $A \stackrel{*}{\Rightarrow} uBv$ is also derivable in G'

In G the derivation is $S \stackrel{*}{\Rightarrow} p\underbrace{A}_{A \to uBv}q \Rightarrow puBvq \stackrel{*}{\Rightarrow} px\underbrace{B}_{B \to w_1}vq \Rightarrow pxw_ivq \stackrel{*}{\Rightarrow} w$

$A \to uBv$

$B \to w_1$

In G' the same string derivation is $S \stackrel{*}{\Rightarrow} pAq \Rightarrow puw_ivq \stackrel{*}{\Rightarrow} pxw_ivq \stackrel{*}{\Rightarrow} w$

$A \to uw_1v$

Leftmost derivation

$u \stackrel{*}{\Rightarrow} x$ where x is a terminal string

Problems:

Remove λ From Grammar

$G_1 = (V, \Sigma, P, S),\ V = \{S, A, B\}, \Sigma = \{a, b\}$

P: $S \rightarrow SaB \mid aB$

$B \rightarrow bB \mid \lambda$

G_1 (with λ rules)

$G_2 = (V, \Sigma\ P, S),\ V = \{S, A, B\}, \Sigma = \{a, b\}$

P: $S \rightarrow SaB \mid SaB \mid aB \mid aB$

$B \rightarrow bB \mid bB \mid \lambda$

P: $S \rightarrow SaB \mid Sa\lambda \mid aB \mid a\lambda$

$B \rightarrow bB \mid b\lambda \mid \lambda$

P: $S \rightarrow SaB \mid Sa \mid aB \mid a$

$B \rightarrow bB \mid b$

Replicate B expressions and replace B with λ then eliminate all λ expressions

$S \Rightarrow SaB$	$S \Rightarrow Sa$
$\Rightarrow SaBaB$	$\Rightarrow Saa$
$\Rightarrow SaBaB$	$\Rightarrow aaa$
$\Rightarrow aBaBaB$	
$\Rightarrow a\lambda aBaB$	
$\Rightarrow aaBaB$	
$\Rightarrow aa\lambda aB$	
$\Rightarrow aaa\lambda$	
$\Rightarrow aaa$	

G_2 (without λ rules)

Remove λ From Grammar

for each rule $A \rightarrow w \in \mathrm{P}$

{if w can be expressed as $w_1 A_1 w_2 A_2 , \ldots , w_k A_k w_{k+1}$

and $A_1, A_2, \ldots, A_k \in Null$ then

add rule $A \rightarrow w_1 w_2 , \ldots , w_k w_{k+1}$ to $\mathrm{P_L}$

}

delete all λ rules

Null is the set of nullable variables in *G*

Problems:

Remove λ From Grammar

$G_1 = (V, \Sigma, P, S)$, $V = \{S, A, B\}$, $\Sigma = \{a, b\}$

P: $S \to aAb$

$A \to aA \mid B$

$B \to bB \mid \lambda$

$L(G_1) = \{a^+ b^+\}$

B is nullable therefore
A is nullable

G_1 (with λ rules)

$S \Rightarrow aAb$	$S \Rightarrow aAb$
$\Rightarrow aaAb$	$\Rightarrow aaAb$
$\Rightarrow aaBb$	$\Rightarrow aaBb$
$\Rightarrow aabBb$	$\Rightarrow aabb$
$\Rightarrow aab\lambda b$	
$\Rightarrow aabb$	

Nullable variable derives the null string

A and B are ***nullable*** variables

$G_2 = (V, \Sigma, P', S)$, $V = \{S, A, B\}$, $\Sigma = \{a, b\}$

P': $S \to aAb \mid aAb$	$S \to aAb \mid ab$
$A \to aA \mid aA \mid B \mid B$	$A \to aA \mid a \mid B$
$B \to bB \mid bB \mid \lambda$	$B \to bB \mid b$

G_2 (without λ rules)

Problems:

Remove λ From Grammar

i. Determine set of ***nullable*** variables

ii. Add rules without (***omitting***) nullable variables

iii. Delete λ rules

Algorithm 4.2.1:

Construct set of ***nullable*** variables: $NULL$ for a CFG $G = (V, \Sigma, P, S)$

$NULL := \{A \mid A \rightarrow \lambda \in \mathrm{P}\}$

While ($NULL$!= $PREV$)

{$PREV$:= $NULL$

for each $A \in \mathrm{V}$

if rule $A \rightarrow w \wedge w \in \mathrm{PREV}^*$ *then* $NULL := NULL \cup \{A\}$

}

permutation of nullable variables

Problems:

Without ***nullable*** variables Grammar is **noncontracting**

(length of Sentential string does not shrink)

Remove λ From Grammar

Example 4.2.1

$G = (V, \Sigma, P, S)$, $V = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$

P: $S \to ACA$

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

$B \to bB \mid b$

$C \to cC \mid \lambda$

$\boldsymbol{A}$ and $\boldsymbol{C}$ are nullable, therefore $\boldsymbol{\lambda}$ is in $\boldsymbol{L(G)}$

Iter	NULL	PREV	*Rule*
0	$\{C\}$		---
1	$\{A, C\}$	$\{C\}$	$A \to C$
2	$\{S, A, C\}$	$\{A, C\}$	$S \to ACA$
3	$\{S, A, C\}$	$\{S, A, C\}$	--

Problems:

Remove λ From Grammar

Example 4.2.2

G = (V, ∑, P, S), $V = \{S, A, B, C\}$, $\sum = \{a, b, c\}$, NULL = $\{C, S, A\}$

P: $S \to ACA \mid ACA \mid ACA \mid ACA \mid ACA \mid ACA \mid ACA$

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

$B \to bB \mid b$

$C \to c\,C \mid c\,C \mid \lambda$

$G_L = (V, \sum, P, S)$, $V = \{S, A, B, C\}$, $\sum = \{a, b, c\}$

P_L: $S \to ACA \mid CA \mid AA \mid AC \mid A \mid C \mid \lambda$

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

$B \to bB \mid$ b

$C \to c\,C \mid c$

λ is in L(G)

Problems:

$S \Rightarrow ACA$	$S \Rightarrow A$
$\Rightarrow aAaCA$	$\Rightarrow aAa$
$\Rightarrow aBaCA$	$\Rightarrow aBa$
$\Rightarrow abaCA$	$\Rightarrow aba$
$\Rightarrow aba\lambda A$	
$\Rightarrow abaA$	
$\Rightarrow abaC$	
$\Rightarrow aba\lambda$	
$\Rightarrow aba$	

Remove λ From Grammar

Example 4.2.3

G = (V, ∑, P, S), $V = \{S, A, B, C\}$, $\sum = \{a, b, c\}$, NULL = $\{A, B, C, S\}$

P: $S \rightarrow ABC$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bB \mid \lambda$

$C \rightarrow c\,C \mid \lambda$

$L(G) = a^*b^*c^*$

$A \rightarrow aA \mid a\underline{A} \mid \lambda$

Problems:

NULL = $\{A, B, C, S\}$, G_L = (V, ∑, P, S), $V = \{S, A, B, C\}$, $\sum = \{a, b, c\}$

P_L: $S \rightarrow ABC \mid AB \mid AC \mid BC \mid A \mid B \mid C \mid \boldsymbol{\lambda}$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

$C \rightarrow c\,C \mid c$

λ in the language

Remove Chained Rules From Grammar

Chains are the result of ***renaming variables***; chained Rules

Rule A → w ∈ V

Rule ∈ $V x V$

$A \overset{*}{\Rightarrow} C$

LHS - RHS

Chains increase the number of steps to create a terminal string

$A \rightarrow aA \mid a \mid B$

$B \rightarrow bB \mid b \mid C$

replace variable B with RHS of B rule

Terminal strings derivable from B are derivable from A

$A \rightarrow aA \mid a \mid bB \mid b \mid C$

$B \rightarrow bB \mid b \mid C$

Problems:

Remove Chained Rules From Grammar

Algorithm 4.3.1 Identifying Chains

input: $G = (V, \Sigma, P, S)$ **essentially noncontracting** CFG (λ *'s* removed)

$CHAIN(A) := \{A\}$ ***/* examine each variable */***

$PREV := \Phi$

While $(CHAIN(A) \mathrel{!=} PREV)$

$\{NEW := CHAIN(A) - PREV$

$PREV :=$ CHAIN (A)

for each $B \in NEW$

$\{$for each $B \rightarrow C$ ***/*** $C \notin CHAIN(A)$ ****/***

$CHAIN(A) := CHAIN(A) \cup \{C\}\ \}$

$\}$

$\}$

Problems:

Remove Chained Rules From Grammar

Example 4.3.1

$G_L = (V, \Sigma, P, S)$, $V = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$

P_L: $S \rightarrow ACA \mid CA \mid AA \mid AC \mid A \mid C \mid \lambda$

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

$B \rightarrow bB \mid b$

$C \rightarrow cC \mid c$

replace variables B and C with RHS of B and C rule

replace variables A, B and C with RHS of A, B and C rule

Iter	PREV	NEW	CHAIN (S)	Rule
0	Φ	$\{S\}$	$\{S\}$	$S \rightarrow A$ $S \rightarrow C$
1	$\{S, A, C\}$	$\{A, C\}$	$\{S, A, C\}$	$A \rightarrow B$
2	$\{S, A, C, B\}$	$\{B\}$	$\{S, A, C, B\}$	
3	$\{S, A, C, B\}$		$\{S, A, C, B\}$	

Iter	PREV	NEW	CHAIN (A)	Rule
0	Φ	$\{A\}$	$\{A\}$	$A \rightarrow B$ $A \rightarrow C$
1	$\{A, B, C\}$	$\{B, C\}$	$\{A, B, C\}$	
2	$\{A, B, C\}$		$\{A, B, C\}$	

Remove Chained Rules From Grammar

Example 4.3.1

$G_C = (V, \Sigma, P, S)$, $V = \{S, A, B, C\}$, $\Sigma = \{a, b, c\}$

P_C: $S \rightarrow ACA \mid CA \mid AA \mid AC \mid aAa \mid aa \mid bB \mid b \mid cC \mid c \mid \lambda$

$A \rightarrow aAa \mid aa \mid bB \mid b \mid cC \mid c$

$B \rightarrow bB \mid b$

$C \rightarrow cC \mid c$

Variable *B not in S rule but in CHAIN*(*S*)

Iter	PREV	NEW	CHAIN (S)	Rule
0	Φ	$\{S\}$	$\{S\}$	$S \rightarrow A$ $S \rightarrow C$
1	$\{S, A, C\}$	$\{A, C\}$	$\{S, A, C\}$	$A \rightarrow B$
2	$\{S, A, C, B\}$	$\{B\}$	$\{S, A, C, B\}$	
3	$\{S, A, C, B\}$		$\{S, A, C, B\}$	

Iter	PREV	NEW	CHAIN (A)	Rule
0	Φ	$\{A\}$	$\{A\}$	$A \rightarrow B$ $A \rightarrow C$
1	$\{A, B, C\}$	$\{B, C\}$	$\{A, B, C\}$	
2	$\{A, B, C\}$		$\{A, B, C\}$	

Remove Chained Rules From Grammar

G_C rules constructed from CHAIN (A) and G rules:

if $B \in$ CHAIN (A) $\wedge$ $B \rightarrow w \in$ P (***original rules***) $\wedge\ w \not\ni$ V
then *Rule* $A \rightarrow w$ is in P_C (***reduced rules***)

replace B with w in rule A

G_C rules are in the form $S \rightarrow \lambda$, $A \rightarrow a$, or $A \rightarrow w \in (V \cup \Sigma)^*$ | *length* $(w) \geq 2$

Problems:

Remove Useless Symbols* From Grammar

G = (V, ∑, P, S), $V = \{S, A, B, C, D, E, F\}$, $\Sigma = \{a, b, c\}$

P: $S \rightarrow AC \mid BS \mid B$

$A \rightarrow aA \mid aF$

$B \rightarrow CF \mid b$

$\boldsymbol{C \rightarrow cC \mid D}$

$\boldsymbol{D \rightarrow aD \mid BD \mid C}$

$\boldsymbol{E \rightarrow aA \mid BSA}$

$F \rightarrow bB \mid b$

Variables not reachable from ***S or***
don't derive a terminal string
Elements in Σ not in a terminal string

Problems:

C, D ***cannot derive a terminal symbol****

E ***is not reachable from S****

Remove Useless Symbols From Grammar

Algorithm 4.4.2: Variables that derive a terminal string

input: $G = (V, \Sigma, P, S)$ CFG

$PREV := \Phi$

$TERM := \{A \mid A \rightarrow w, w \in \Sigma^*\}$

While ($TERM\ != PREV$)

  { $PREV := TERM$

    for each $A \in V - PREV$

      {if $A \rightarrow w \in (TERM \cup \Sigma)^*$ *then*

            /* w is some permutation of $TERM$ and Σ */

        $TERM := TERM \cup \{A\}$ }

      }

  }

  /* **Useless Variables** $= V - TERM$ */

Problems:

Remove Useless Symbols From Grammar

Theorem 4.4.3

For $G = (V, \Sigma, P, S)$ a *CFG,* there is a an algorithm to construct

$G_T = (V_T \Sigma_T P_T S) \mid$

$L(G_T) = L(G)$

$V_T = TERM \subseteq V$

$P_T = \{ A \rightarrow w, A \in TERM, w \in (TERM \cup \Sigma)^* \}$

$\Sigma_T = \{ a \in \Sigma \mid a$ occurs in rules $P_T \}$

Problems:

Remove Useless Symbols From Grammar

Example 4.4.1

G = (V, ∑, P, S), $V = \{S, A, B, C, D, E, F\}, \Sigma = \{a, b, c\}$

P: $S \to AC \mid BS \mid B$

$A \to aA \mid aF$

$B \to CF \mid b$

$C \to c\,C \mid D$

$D \to aD \mid BD \mid C$

$E \to aA \mid BSA$

$F \to bB \mid b$

$BD \notin (TERM \cup \Sigma)^*$

Iter	PREV	TERM	Rule
0	Φ	$\{B, F\}$	$B \to b$ $F \to b$
1	$\{B, F\}$	$\{B, F, S, A\}$	$S \to B$ $A \to aF$
2	$\{B, F, S, A\}$	$\{B, F, S, A, E\}$	$E \to aA$
3	$\{B, F, S, A, E\}$	$\{B, F, S, A, E\}$	

$TERM = \{B, F, S, A, E\}$

$UselessVariables = V - TERM = \{S, A, B, C, D, E, F\} - TERM = \{C, D\}$

Problems:

Remove Useless Symbols From Grammar

Example 4.4.1

$G_T = (V_T, \sum_T, P_T, S)$,

$V_T = \{S, A, B, E, F\}, \sum_T = \{a, b\}$

P_T: $S \rightarrow BS \mid B$

$A \rightarrow aA \mid aF$

$B \rightarrow b$

$E \rightarrow aA \mid BSA$

$F \rightarrow bB \mid b$

$UselessVariables = \{C, D\}$

$\boldsymbol{c}$ eliminated

$\boldsymbol{S \rightarrow AC}$ eliminated

$\boldsymbol{B \rightarrow CF}$ eliminated

Iter	PREV	TERM	Rule
0	Φ	$\{B, F\}$	$B \rightarrow b$ $F \rightarrow b$
1	$\{B, F\}$	$\{B, F, S, A\}$	$S \rightarrow B$ $A \rightarrow aF$
2	$\{B, F, S, A\}$	$\{B, F, S, A, E\}$	$E \rightarrow aA$
3	$\{B, F, S, A, E\}$	$\{B, F, S, A, E\}$	

Remove Useless Symbols From Grammar

Algorithm 4.4.4: Variables reachable from S

input: $G = (V, \Sigma, P, S)$ CFG

$REACH := \{S\}$

$PREV := \Phi$

While ($REACH$!= $PREV$)

{NEW := $REACH$ - $PREV$

$PREV$:= $REACH$

for each $A \in NEW$

{for each $A \rightarrow w \in ((V\text{-}REACH) \cup \Sigma)^*$

Permutation of unreachable Variables and Σ

/* add **all** variables in w to $REACH$ */}

}

$Non\text{-}Reachable\ Variables = V - REACH$

Remove Useless Symbols From Grammar

Theorem 4.4.6

For $G = (V, \Sigma, P, S)$ a *CFG*, there is a an algorithm to construct

$G_U = (V_U, \Sigma_U, P_U, S) \mid$

G_U has no useless symbols

$L(G_U) = L(G)$

$V_U = REACH \subseteq V$

$P_U = \{ A \rightarrow w \in P, A \in REACH, w \in (REACH \cup \Sigma)^* \}$

$\Sigma_U = \{ a \in \Sigma \mid a$ occurs in RHS of a rule in $P_U \}$

Problems:

Remove Useless Symbols From Grammar

Example 4.4.2

$G_T = (V_T, \Sigma_T, P_T, S)$,

$V_T = \{S, A, B, E, F\}, \Sigma = \{a, b\}$

P_T: $S \to BS \mid B$

$A \to aA \mid aF$

$B \to b$

$E \to aA \mid BSA$

$F \to bB \mid b$

$G_U = (V_U, \Sigma_U, P_U, S)$,

$V_U = \{S, B\}, \Sigma_U = \{b\}$

P_U: $S \to BS \mid B$

$B \to b$

Iter	REACH	PREV	NEW	Rule
0	$\{S\}$	Φ		
1	$\{S, B\}$	$\{S\}$	$\{S\}$	$S \to B$
2	$\{S, B\}$	$\{S, B\}$	$\{B\}$	
3	$\{S, B\}$	$\{S, B\}$		

$NonReachableVariables =$

$$V - REACH = \{S, A, B, E, F\} - \{S, B\} = \{A, F, E\}$$

Remove Useless Symbols From Grammar

Example 4.4.3

$G = (V, \Sigma, P, S)$, $V = \{S, A, B\}$, $\Sigma = \{a, b\}$

P: $S \rightarrow a \mid AB$

$A \rightarrow b$

Remove variables that do not derive terminal symbols

P_T: $S \rightarrow a$

$A \rightarrow b$

Remove unreachable symbols

P_U: $S \rightarrow a$

Remove unreachable symbols

P_U: $S \rightarrow a \mid AB$

$A \rightarrow b$

Remove variables that do not derive terminal symbols

P_T: $S \rightarrow a$

$A \rightarrow b$

Order matters!!!

Chomsky Normal Form

Definition 4.5.1

$G = (V, \Sigma, P, S)$ a *CFG* is in Chomsky normal form when rules are in the form:

$A \rightarrow BC$, $A \rightarrow a \in \Sigma$, or $A \rightarrow \lambda$, where $B, C \in V - \{S\}$

$P \subseteq \{S\}\text{x}\{\lambda\} \cup V\text{x}\Sigma \cup V\text{x}(V\text{-}\{S\})^2$

$A \rightarrow$ concatenated variables

$A \rightarrow \Sigma$

$S \rightarrow \lambda$

Example

$A \rightarrow bDcF$ is replaced with:

$A \rightarrow B'DC'F$

$B' \rightarrow b$

$C' \rightarrow c$

and $A \rightarrow B'DC'F$ *is* replaced with:

$A \rightarrow B'T_1$

$T_1 \rightarrow DT_2$

$T_2 \rightarrow C'F$

Problems:

Chomsky Normal Form

$$Replace: \quad A \rightarrow X_1 X_2 \; . \; . \; . \; X_n$$

$$A \rightarrow X_1 B_1$$

$$B_1 \rightarrow X_2 B_2$$

$$.$$

$$.$$

$$.$$

$$B_{n-3} \rightarrow X_{n-2} B_{n-2}$$

$$B_{n-2} \rightarrow X_{n-1} X_n$$

Left to right processing

$B_1, B_2, \; . \; . \; . \; , B_{n-2}$ *are new variables (non- terminal symbols)*

Problems:

Chomsky Normal Form

Example 4.5.1

For $G = (V, \sum, P, S)$, $V = \{A, B, C\}, \sum = \{a, b, c\}$

a *CFG* that satisfies :

- ***start symbol conditions***
- ***λ rules***
- ***chain rules***
- ***eliminated useless symbols***

$V = \{S, A, B, C\}, \sum = \{a, b, c\}$

$$
\begin{aligned}
P: \quad & S \rightarrow aABC \mid a \\
& A \rightarrow aA \mid a \\
& B \rightarrow bcB \mid bc \\
& C \rightarrow c\,C \mid c
\end{aligned}
$$

$$
\begin{aligned}
P': \quad & S \rightarrow A'T_1 \mid a \\
& A' \rightarrow a \\
& T_1 \rightarrow AT_2 \\
& T_2 \rightarrow BC \\
& A \rightarrow A'A \mid a \\
& B' \rightarrow b \\
& C' \rightarrow c \\
& B \rightarrow B'T_3 \mid B'C' \\
& T_3 \rightarrow C'B \\
& C \rightarrow C'C \mid c
\end{aligned}
$$

Problems:

Chomsky Normal Form

Example 4.5.2

Rule: $X \rightarrow aXb \mid ab$

Generates strings $\{a^i b^i, i \geq 1\}$

$$S \rightarrow X$$
$$X \rightarrow aXb \mid ab$$

Eliminate X Chain

$$S \rightarrow aXb \mid ab$$
$$X \rightarrow aXb \mid ab$$

$$S \rightarrow AT \mid AB$$
$$X \rightarrow AT \mid AB$$
$$A \rightarrow a$$
$$B \rightarrow b$$
$$T \rightarrow XB$$

Problems:

CYK Algorithm

string $x_{1,2}$ string $x_{3,4}$

a *a* *a* *b* *b* *b*

aa *aa* *ab* *bb* *bb*

aaa *aab* *abb* *bbb*

aaab *aabb* *abbb*

aaabb *aabbb*

aaabbb

string $x_{1,4}$ string $x_{3,6}$

Example 4.5.2

string x_i

set of variables that derives string $\boldsymbol{X}_{i,i}$

Step 0:

If $A \rightarrow x_i$ *then* $X_{ii} = \{A\}$

Step 1:

If $A \rightarrow BC$, $B \Rightarrow x_i$ $(B \in X_{ii})$,

$C \Rightarrow x_{i+1}$ $(C \in X_{i+1,\, i+1})$ *then*

$$X_{i,i+1} = X_{i,i+1} \cup \{A\}$$

Step t: $t = 3; i = 1, \ldots, t;$ $(i \le k < i + t)$

If $A \rightarrow BC$, $B \Rightarrow x_{i,k}$ $(B \in X_{i,k})$,

$C \Rightarrow x_{k+1,\, i+t}$ $(C \in X_{k+1,\, i+t})$ *then*

$$X_{i,\, i+t} = X_{i,\, i+t} \cup \{A\}$$

Problems:

CYK Algorithm

For Grammar

$G = (V, \Sigma, P, S)$, $V = \{A, B, T, X, S\}$, $\Sigma = \{a, b\}$

$S \rightarrow AT \mid AB$

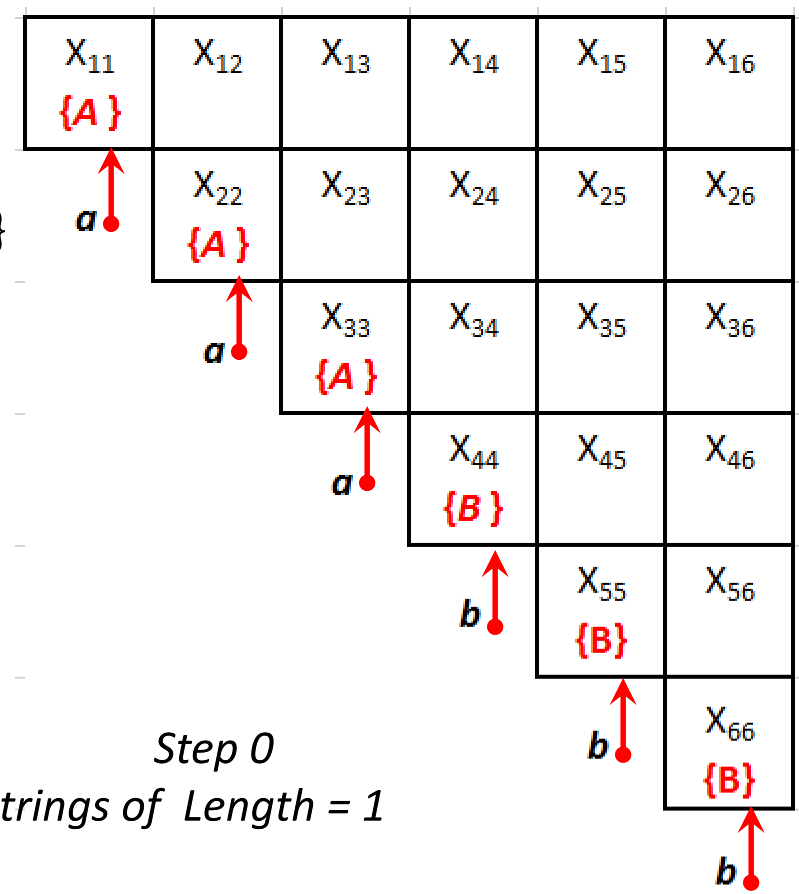
$X \rightarrow AT \mid AB$

$A \rightarrow a$

$B \rightarrow b$

$T \rightarrow XB$

Validate string ***aaabbb***



Step 0

Strings of Length = 1

Problems:

CYK Algorithm

For Grammar

$G = (V, \Sigma, P, S)$, $V = \{A, B, T, S, X\}$, $\Sigma = \{a, b\}$

$S \rightarrow AT \mid AB$

$X \rightarrow AT \mid AB$

$A \rightarrow a$

$B \rightarrow b$

$T \rightarrow XB$

Validate string ***aaabbb***

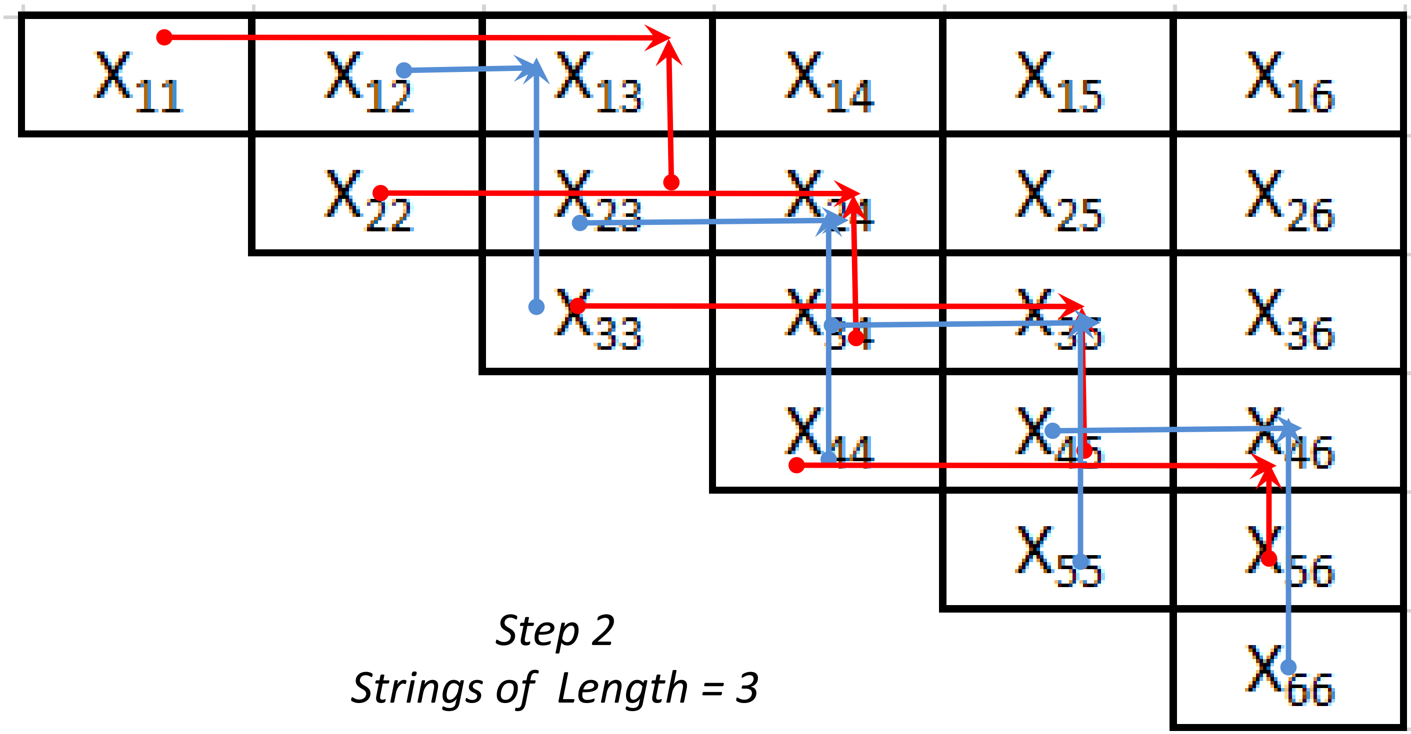
X_{11} {A}	X_{12} Φ	X_{13}	X_{14}	X_{15}	X_{16}
	X_{22} {A}	X_{23} Φ	X_{24}	X_{25}	X_{26}
		X_{33} {A}	X_{34} {S, X}	X_{35}	X_{36}
			X_{44} {B}	X_{45} Φ	X_{46}
				X_{55} {B}	X_{56} Φ
					X_{66} {B}

Step 1

Strings of Length = 2

Problems:

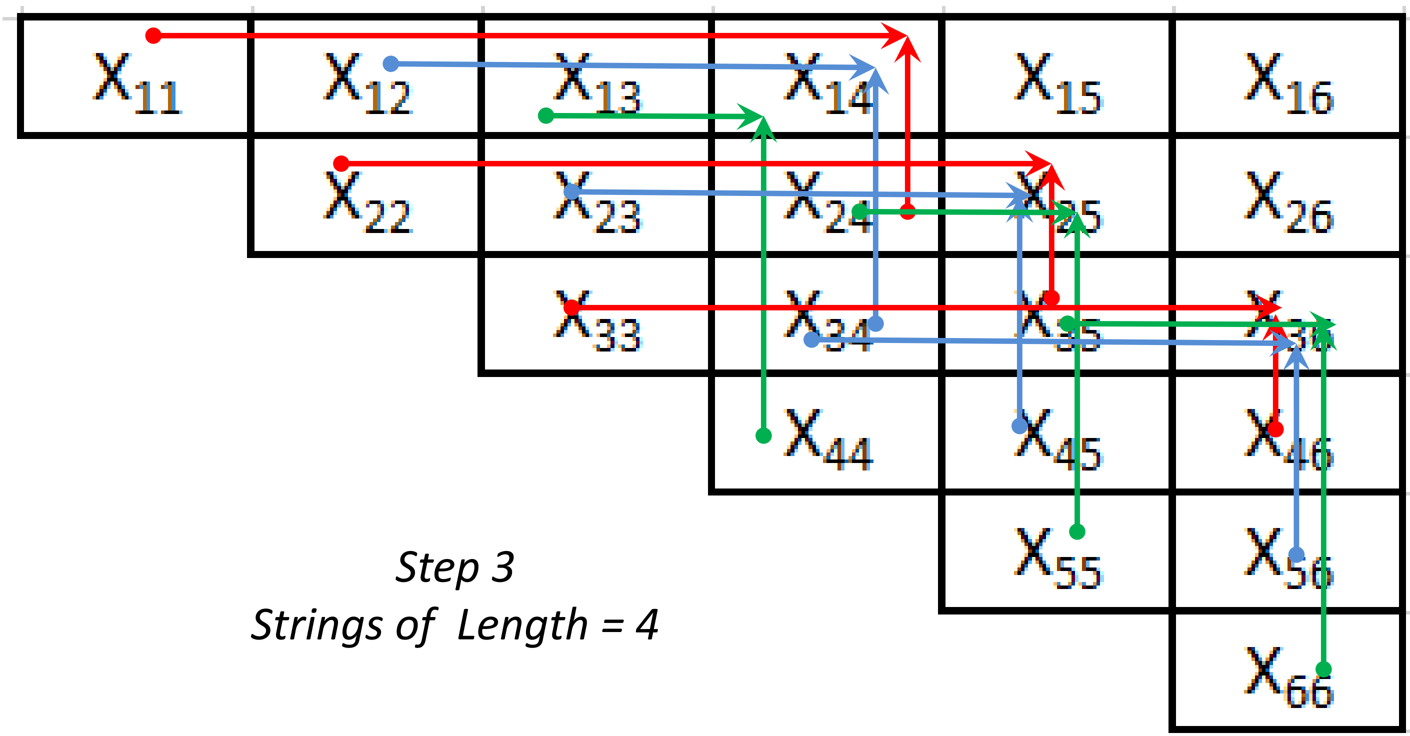
CYK Algorithm



Step 2
Strings of Length = 3

Problems:

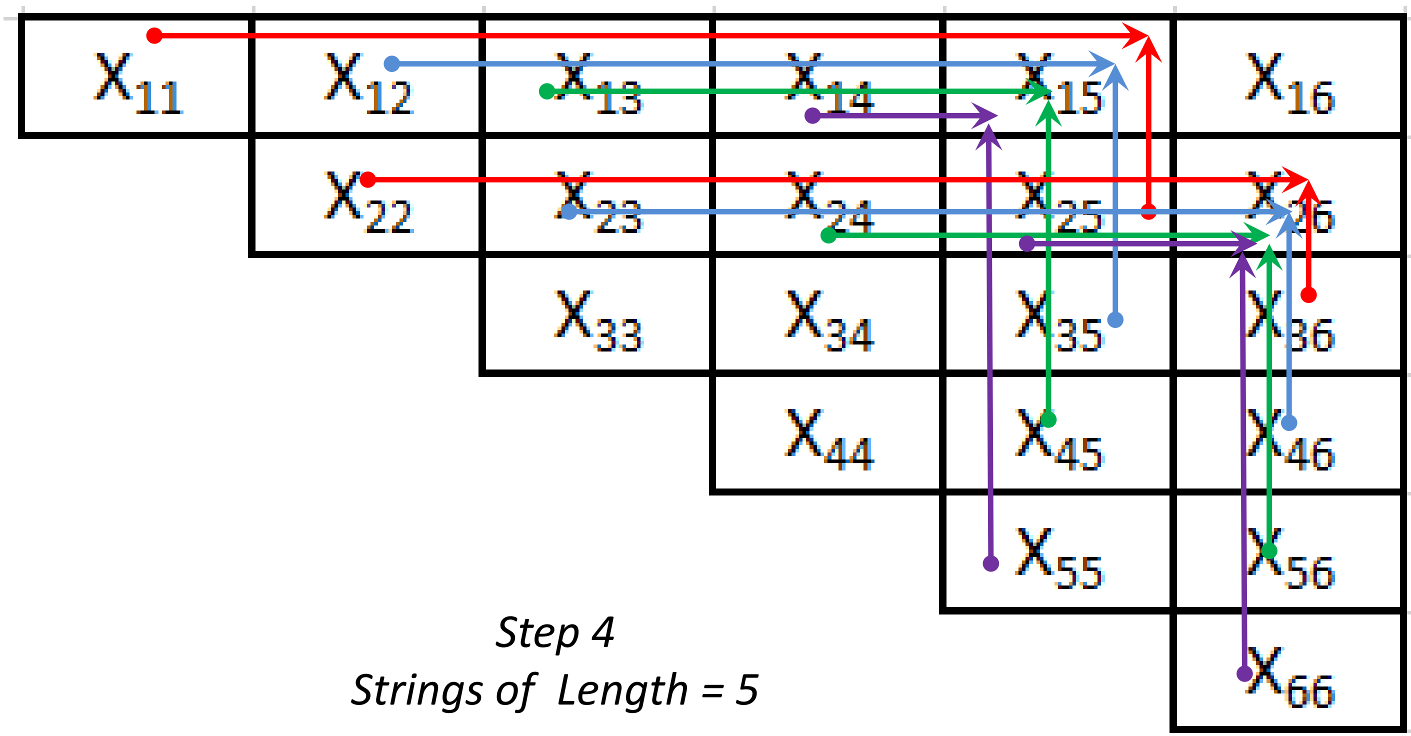
CYK Algorithm



Step 3
Strings of Length = 4

Problems:

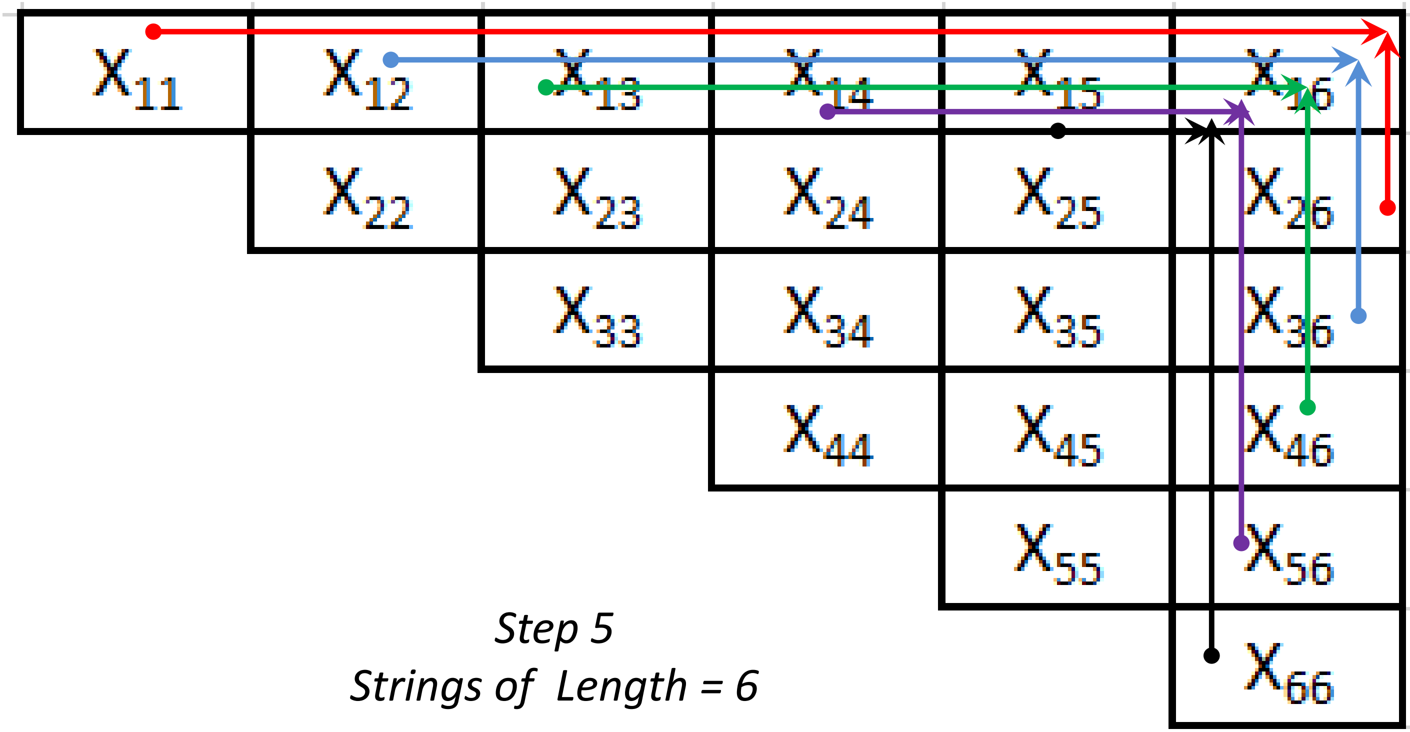
CYK Algorithm



Step 4
Strings of Length = 5

Problems:

CYK Algorithm



Step 5
Strings of Length = 6

Problems:

CYK Algorithm

For Chomsky Normal Form Grammar

for valid string, ***S*** *must be in set* $\boldsymbol{X_{16}}$

$G = (V, \Sigma, P, S)$, $V = \{A, B, T, X, S\}$, $\Sigma = \{a, b\}$

$S \rightarrow AT \mid AB$

$X \rightarrow AT \mid AB$

$A \rightarrow a$

$B \rightarrow b$

$T \rightarrow XB$

X_{11} {A}	X_{12} Φ	X_{13} Φ	X_{14} Φ	X_{15} Φ	X_{16} {S, X}
	X_{22} {A}	X_{23} Φ	X_{24} Φ	X_{25} {S, X}	X_{26} {T}
		X_{33} {A}	X_{34} {S, X}	X_{35} {T}	X_{36} Φ
			X_{44} {B}	X_{45} Φ	X_{46} Φ
				X_{55} {B}	X_{56} Φ
					X_{66} {B}

Validate string ***aaabbb***

Problems:

Remove Left Recursion

$G_1: S \to Aa$

$A \to Aa \mid b$

Generates string on right

$S \Rightarrow Aa$
$\Rightarrow Aaa$
$\Rightarrow Aaaa$
$\Rightarrow baaa$

$G_2: S \to aAb$

$A \to aAb \mid \lambda$

Generates string on both sides

only ***terminal symbols*** before first variable

*Right Recursion builds (**terminal prefix**) string on left*

Problems:

Remove Direct Left Recursion

Divide Recursive rule into:

Left Recursive terms

replace $A \rightarrow Au_1 \mid Au_2 \mid \ldots \mid Au_j$

with $Z \rightarrow u_1Z \mid u_2Z \mid \ldots \mid u_jZ \mid u_1 \mid u_2 \mid \ldots \mid u_j$

Terms with first RHS symbol not A

replace $A \rightarrow v_1 \mid v_2 \mid \ldots \mid v_j$

with $A \rightarrow v_1Z \mid v_2Z \mid \ldots \mid v_kZ \mid v_1 \mid v_2 \mid \ldots \mid v_k$

$A \rightarrow Aa \mid Aab \mid bb \mid b$ *is replaced by*

$A \rightarrow bbZ \mid bZ \mid bb \mid b$

$Z \rightarrow aZ \mid abZ \mid a \mid ab$

*build (**terminal prefix**) string on left first*

Problems:

Remove Indirect Left Recursion

Does not solve indirect left recursion Problem

$A \rightarrow Bu$

$B \rightarrow Av$

$A \Rightarrow Bu$
$\Rightarrow Avu$
$\Rightarrow Buvu$
$\Rightarrow Avuvu$

Greibach Normal Form solves Left Recursion - Direct and Indirect

$G = \{V, \sum, P, S\}$ is Greibach Normal Form when:

$A \rightarrow aA_1A_2 \ldots A_n$

$A \rightarrow a$

or

$A \rightarrow \lambda$

$a \in \sum \wedge \; A_i \in V - \{S\}$

Problems: