Theory of Automata
Greibach Normal Form

Week 9

Contents

- Definition – CNF
- Conversion of a Chomsky normal form grammar to Greibach normal form

Definition

- A CFG is in Greibach normal form if each rule has one these forms:

i. $A \rightarrow aA_1A_2\ldots A_n$

ii. $A \rightarrow a$

iii. $S \rightarrow \lambda$

where $a \in \Sigma$ and $A_i \in V - \{S\}$ for $i = 1, 2, \ldots, n$

Definition

- A CFG is in Chomsky normal form if each rule has one these forms:

i. $A \rightarrow BC$

ii. $A \rightarrow a$

iii. $S \rightarrow \lambda$

where $B, C \in V - \{S\}$

Conversion

- Convert from Chomsky to Greibach in two steps:

1. From Chomsky to intermediate grammar
   a. Eliminate direct left recursion
   b. Use $A \rightarrow uBv$ rules transformations to improve references (explained later)
2. From intermediate grammar into Greibach

Eliminate direct left recursion

- Before

  $A \rightarrow A\underline{a} \mid \boldsymbol{b}$

- After

  $A \rightarrow \boldsymbol{b}Z \mid \boldsymbol{b}$

  $Z \rightarrow \underline{a}Z \mid \underline{a}$

- Remove the rule with direct left recursion, and create a new one with recursion on the right

Eliminate direct left recursion

- Before

  $A \rightarrow A\underline{a} \mid A\underline{b} \mid \boldsymbol{b} \mid \boldsymbol{c}$

- After

  $A \rightarrow \boldsymbol{b}\underline{Z} \mid \boldsymbol{c}\underline{Z} \mid \boldsymbol{b} \mid \boldsymbol{c}$

  $Z \rightarrow \underline{a}Z \mid \underline{b}Z \mid \underline{a} \mid \underline{b}$

- Remove the rules with direct left recursion, and create new ones with recursion on the right

Eliminate direct left recursion

- Before

$A \rightarrow A\underline{B} \mid \boldsymbol{BA} \mid \boldsymbol{a}$

$B \rightarrow b \mid c$

- After

$A \rightarrow \boldsymbol{BA}\underline{Z} \mid \boldsymbol{a}\underline{Z} \mid \boldsymbol{BA} \mid \boldsymbol{a}$

$Z \rightarrow \underline{B}Z \mid \underline{B}$

$B \rightarrow b \mid c$

Transform $A \rightarrow uBv$ rules

- Before

  $A \rightarrow uBb$

  $B \rightarrow w_1 \mid w_2 \mid \ldots \mid w_n$

- After

  Add $A \rightarrow uw_1b \mid uw_2b \mid \ldots \mid uw_nb$

  Delete $A \rightarrow uBb$

Conversion: Step 1

- Goal: construct intermediate grammar in this format

  i. $A \rightarrow aw$

  ii. $A \rightarrow Bw$

  iii. $S \rightarrow \lambda$

  where $w \in V^*$ and B comes after A

Conversion: Step 1

- Assign a number to all variables starting with S, which gets 1
- Transform each rule following the order according to given number from lowest to highest
  - Eliminate direct left recursion
  - If RHS of rule starts with variable with lower order, apply $A \rightarrow uBb$ transformation to fix it

Conversion: Step 2

- Goal: construct Greibach grammar out of intermediate grammar from step 1
- Fix $A \rightarrow Bw$ rules into $A \rightarrow aw$ format
  - After step 1, last original variable should have all its rules starting with a terminal
  - Working from bottom to top, fix all original variables using $A \rightarrow uBb$ transformation technique, so all rules become $A \rightarrow aw$
- Fix introduced recursive rules same way

Conversion Example

- Convert the following grammar from Chomsky normal form, into Greibach normal form

1. $S \rightarrow AB \mid \lambda$
2. $A \rightarrow AB \mid CB \mid a$
3. $B \rightarrow AB \mid b$
4. $C \rightarrow AC \mid c$

Conversion Strategy

- Goal: transform all rules which RHS does not start with a terminal
- Apply two steps conversion
- Work rules in sequence, eliminating direct left recursion, and enforcing variable reference to higher given number
- Fix all original rules, then new ones

Step 1: S rules

- Starting with S since it has a value to of 1
- $S \rightarrow AB \mid \lambda$
- S rules comply with two required conditions
  - There is no direct left recursion
  - Referenced rules A and B have a given number higher than 1. A corresponds to 2 and B to 3.

Step 1: A rules

- $A \rightarrow A\underline{B} \mid \boldsymbol{CB} \mid \boldsymbol{a}$
- Direct left recursive rule $A \rightarrow AB$ needs to be fixed. Other A rules are fine
- Apply direct left recursion transformation

  $A \rightarrow \boldsymbol{CB}\underline{R_1} \mid \boldsymbol{a}\underline{R_1} \mid \boldsymbol{CB} \mid \boldsymbol{a}$

  $R_1 \rightarrow \underline{B}R_1 \mid \underline{B}$

Step 1: B rules

- $B \rightarrow \underline{A}\boldsymbol{B} \mid b$
- $B \rightarrow AB$ rule needs to be fixed since B corresponds to 3 and A to 2. B rules can only have on their RHS variables with number equal or higher. Use $A \rightarrow uBb$ transformation technique
- $B \rightarrow \underline{CBR_1}\boldsymbol{B} \mid \underline{aR_1}\boldsymbol{B} \mid \underline{CB}\boldsymbol{B} \mid \underline{a}\boldsymbol{B} \mid b$

Step 1: C rules

- $C \rightarrow \underline{A}\boldsymbol{C} \mid c$
- $C \rightarrow AC$ rule needs to be fixed since C corresponds to 4 and A to 2. Use same $A \rightarrow uBb$ transformation technique
- $C \rightarrow \underline{CBR_1}\boldsymbol{C} \mid \underline{aR_1}\boldsymbol{C} \mid \underline{CB}\boldsymbol{C} \mid \underline{a}\boldsymbol{C} \mid c$
- Now variable references are fine according to given number, but we introduced direct left recursion in two rules…

Step 1: C rules

- $C \rightarrow C\underline{BR_1C} \mid \boldsymbol{aR_1C} \mid C\underline{BC} \mid \boldsymbol{aC} \mid \boldsymbol{c}$
- Eliminate direct left recursion

  $C \rightarrow \boldsymbol{aR_1C}\underline{R_2} \mid \boldsymbol{aC}\underline{R_2} \mid \boldsymbol{c}\underline{R_2} \mid \boldsymbol{aR_1C} \mid \boldsymbol{aC} \mid \boldsymbol{c}$

  $R_2 \rightarrow \underline{BR_1C}R_2 \mid \underline{BC}R_2 \mid \underline{BR_1C} \mid \underline{BC}$

Step 1: Intermediate grammar

- $S \rightarrow AB \mid \lambda$
- $A \rightarrow CBR_1 \mid aR_1 \mid CB \mid a$
- $B \rightarrow CBR_1B \mid aR_1B \mid CBB \mid aB \mid b$
- $C \rightarrow aR_1CR_2 \mid aCR_2 \mid cR_2 \mid aR_1C \mid aC \mid c$
- $R_1 \rightarrow BR_1 \mid B$
- $R_2 \rightarrow BR_1CR_2 \mid BCR_2 \mid BR_1C \mid BC$

Step 2: Fix starting symbol

- Rules S, A, B and C don't have direct left recursion, and RHS variables are of higher number
- All C rules start with terminal symbol
- Proceed to fix rules B, A and S in bottom-up order, so they start with terminal symbol.
- Use $A \rightarrow uBb$ transformation technique

Step 2: Fixing B rules

- Before

  $B \rightarrow \underline{C}\boldsymbol{BR_1B} \mid aR_1B \mid \underline{C}\boldsymbol{BB} \mid aB \mid b$

- After

  $B \rightarrow aR_1B \mid aB \mid b$

  $B \rightarrow \underline{aR_1CR_2}\boldsymbol{BR_1B} \mid \underline{aCR_2}\boldsymbol{BR_1B} \mid \underline{cR_2}\boldsymbol{BR_1B} \mid \underline{aR_1C}\boldsymbol{BR_1B} \mid \underline{aC}\boldsymbol{BR_1B} \mid \underline{c}\boldsymbol{BR_1B}$

  $B \rightarrow \underline{aR_1CR_2}\boldsymbol{BB} \mid \underline{aCR_2}\boldsymbol{BB} \mid \underline{cR_2}\boldsymbol{BB} \mid \underline{aR_1C}\boldsymbol{BB} \mid \underline{aC}\boldsymbol{BB} \mid \underline{c}\boldsymbol{BB}$

Step 2: Fixing A rules

- Before

  $A \rightarrow \underline{C}\boldsymbol{BR_1} \mid aR_1 \mid \underline{C}\boldsymbol{B} \mid a$

- After

  $A \rightarrow aR_1 \mid a$

  $A \rightarrow \underline{aR_1CR_2}\boldsymbol{BR_1} \mid \underline{aCR_2}\boldsymbol{BR_1} \mid \underline{cR_2}\boldsymbol{BR_1} \mid \underline{aR_1C}\boldsymbol{BR_1} \mid \underline{aC}\boldsymbol{BR_1} \mid \underline{c}\boldsymbol{BR_1}$

  $A \rightarrow \underline{aR_1CR_2}\boldsymbol{B} \mid \underline{aCR_2}\boldsymbol{B} \mid \underline{cR_2}\boldsymbol{B} \mid \underline{aR_1C}\boldsymbol{B} \mid \underline{aC}\boldsymbol{B} \mid \underline{c}\boldsymbol{B}$

Step 2: Fixing S rules

- Before

  $S \rightarrow \underline{A}\boldsymbol{B} \mid \lambda$

- After

  $S \rightarrow \lambda$

  $S \rightarrow \underline{aR_1}\boldsymbol{B} \mid \underline{a}\boldsymbol{B}$

  $S \rightarrow \underline{aR_1CR_2BR_1}\boldsymbol{B} \mid \underline{aCR_2BR_1}\boldsymbol{B} \mid \underline{cR_2BR_1}\boldsymbol{B} \mid \underline{aR_1CBR_1}\boldsymbol{B} \mid \underline{aCBR_1}\boldsymbol{B} \mid \underline{cBR_1}\boldsymbol{B}$

  $S \rightarrow \underline{aR_1CR_2B}\boldsymbol{B} \mid \underline{aCR_2B}\boldsymbol{B} \mid \underline{cR_2B}\boldsymbol{B} \mid \underline{aR_1CB}\boldsymbol{B} \mid \underline{aCB}\boldsymbol{B} \mid \underline{cB}\boldsymbol{B}$

Step 2: Complete conversion

- All original rules S, A, B and C are fully converted now
- New recursive rules need to be converted next

  $R_1 \rightarrow BR_1 \mid B$

  $R_2 \rightarrow BR_1CR_2 \mid BCR_2 \mid BR_1C \mid BC$
- Use same $A \rightarrow uBb$ transformation technique replacing starting variable B

Conclusions

- After conversion, since *B* has 15 rules, and R_1 references *B* twice, R_1 ends with 30 rules
- Similar for R_2 which references *B* four times. Therefore, R_2 ends with 60 rules
- All rules start with a terminal symbol (with the exception of $S \rightarrow \lambda$)
- Parsing algorithms top-down or bottom-up would complete on a grammar converted to Greibach normal form

Comparison of Normal forms

$S \rightarrow Sa\mathbf{B} \mid aB$

$B \rightarrow bB \mid \lambda$

- Adding a non-recursive start symbol *S'* and removing λ and chain rules yields

$S' \rightarrow Sa\mathbf{B} \mid Sa \mid aB \mid a$

$S \rightarrow Sa\mathbf{B} \mid Sa \mid aB \mid a$

$B \rightarrow bB \mid b$

- Chomsky Normal form is obtained as:

$S' \rightarrow ST \mid SA \mid AB \mid a$

$S \rightarrow ST \mid SA \mid AB \mid a$

$B \rightarrow CB \mid b$

$T \rightarrow CB$

$A \rightarrow a$

$C \rightarrow b$

- Greibach Normal form is obtained as:

$S' \rightarrow aBZT \mid aZT \mid aBT \mid aT \mid aBZA \mid aZA \mid aBA \mid aA \mid aB \mid a$

$S \rightarrow aBZ \mid aZ \mid aB \mid a$

$B \rightarrow bB \mid b$

$T \rightarrow bB$

$A \rightarrow a$

$C \rightarrow b$

$Z \rightarrow aBZ \mid aZ \mid aB \mid a$

abaaba

G

$S \Rightarrow SaB$

$\Rightarrow SaBaB$

$\Rightarrow SaBaBaB$

$\Rightarrow aBaBaBaB$

$\Rightarrow abaBaBaBaB$

$\Rightarrow abaBaBaB$

$\Rightarrow abaaBaB$

$\Rightarrow abaabBaB$

$\Rightarrow abaabaB$

$\Rightarrow abaaba$

CNF

$S \Rightarrow SA$

$\Rightarrow STA$

$\Rightarrow SATA$

$\Rightarrow ABATA$

$\Rightarrow aBATA$

$\Rightarrow abATA$

$\Rightarrow abaTA$

$\Rightarrow abaABA$

$\Rightarrow abaaBA$

$\Rightarrow abaabA$

$\Rightarrow abaaba$

GNF

$S \Rightarrow aBZA$

$\Rightarrow abZA$

$\Rightarrow abaZA$

$\Rightarrow abaaBA$

$\Rightarrow abaabA$

$\Rightarrow abaaba$

Examples

- Convert to GNF
- S->aSX|b
- X->Xb|a