COURSE: THEORY OF AUTOMATA COMPUTATION

TOPICS TO BE COVERED

- Greibach Normal Form
- 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. $A \to aA_1A_2 \dots A_n$

ii. $A \to a$

iii. $S \to \lambda$

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

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

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

- After

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

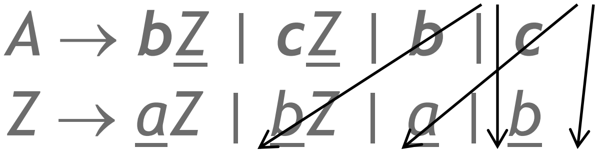
- Remove the rule with direct left recursion, and create a new one with recursion on the right

ELIMINATE DIRECT LEFT RECURSION

- Before

$A \to A\underline{a} \mid A\underline{b} \mid \mathbf{b} \mid \mathbf{c}$

- After



$A \to \mathbf{b}\underline{Z} \mid \mathbf{c}\underline{Z} \mid \mathbf{b} \mid \mathbf{c}$

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

- 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

  $A \rightarrow uBb$

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

- After

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

  Delete $A \rightarrow uBb$

CONVERSION: STEP 1

- Goal: construct intermediate grammar in this format

i. $A \to aw$

ii. $A \to Bw$

iii. $S \to \lambda$

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 $A \to Bw$ rules into $A \to aw$ 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 $A \to uBb$ transformation technique, so all rules become $A \to aw$
- 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

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

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

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

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

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

STEP 1: C RULES

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

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

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

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 $A \to uBb$ 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

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

  $B \rightarrow \underline{aR_1}\underline{CR_2}\boldsymbol{BR_1B} \mid \underline{aCR_2}\boldsymbol{BR_1B} \mid \underline{cR_2}\boldsymbol{BR_1B} \mid \underline{aR_1}\underline{C}\boldsymbol{BR_1B} \mid \underline{aC}\boldsymbol{BR_1B} \mid \underline{c}\boldsymbol{BR_1B}$

  $B \rightarrow \underline{aR_1}\underline{CR_2}\boldsymbol{BB} \mid \underline{aCR_2}\boldsymbol{BB} \mid \underline{cR_2}\boldsymbol{BB} \mid \underline{aR_1}\underline{C}\boldsymbol{BB} \mid \underline{aC}\boldsymbol{BB} \mid \underline{c}\boldsymbol{BB}$

STEP 2: FIXING A RULES

- Before

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

- After

$A \rightarrow aR_1 \mid a$

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

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

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