補充資料 II
(Context Free Grammar)

資科系
林偉川

CFG Important Rules

Rule 1. Substitution and elimination

CFG G=(V,T,S,P) , where P has the forms

$A \rightarrow x_1 B x_2$

$B \rightarrow y_1 \mid y_2 \mid y_3 \mid \ldots \mid y_n$

Can be changed to

$A \rightarrow x_1 y_1 x_2 \mid x_1 y_2 x_2 \mid \ldots \mid x_1 y_n x_2$

2

Example

G=({A,B},{a,b,c}, A, P), where P are given by

A → a | aaA | abBc

B → abbA | b

can be changed to

A → a | aaA | abbc | ababbAc

Ex.

A => aaA => aaabBc => aaabbc (in original)

A => aaA => aaabbc (in modified)

3

CFG Important Rules

Rule 2. Eliminate the Left Recursion

CFG G=(V,T,S,P) , where P has the forms

$A \rightarrow Ax_1 \mid Ax_2 \mid Ax_3 \mid \ldots \mid Ax_n$

$A \rightarrow y_1 \mid y_2 \mid y_3 \mid \ldots \mid y_m$

Can be changed to

$A \rightarrow y_i \mid y_i Z$, i=1,2,…,m

$Z \rightarrow x_i \mid x_i Z$, i=1,2,…,m

4

Example

G=({A,B},{a,b,c}, A, P), where P are given by

$A \rightarrow Aa \mid aBc \mid \lambda$

$B \rightarrow Bb \mid ba$ (assume x_1=a, y_1=aBc, y_2= λ)

can be changed to

$A \rightarrow aBc \mid aBcZ \mid Z \mid \lambda$

$Z \rightarrow a \mid aZ$

$B \rightarrow Bb \mid ba$ (in original)

5

Example(continued)

Finally can be futher changed to

$A \rightarrow aBc \mid aBcZ \mid Z \mid \lambda$

$Z \rightarrow a \mid aZ$

$B \rightarrow ba \mid baY$

$Y \rightarrow b \mid bY$

6

CFG Important Rules

Rule 3. Eliminate the useless production(Rule)

Useless Non-terminal:

a. Cannot reach from the start symbol

b. Cannot derive a terminal string

Given a CFG G=(V,T,S,P) , $\forall V_i \in V$, V_i is not useless symbol ∋ G is a useful grammar

7

Example

G=({S,A,B},{a,b}, S, P), where P are given by

$S \rightarrow A$

$A \rightarrow aA \mid \lambda$

$B \rightarrow bA$

can be changed to

$S \rightarrow A$

$A \rightarrow aA \mid \lambda$

8

Example

G=({S,A,B,C},{a,b}, S, P), where P are given by

S → aS | A | C

A → a

B → aa (cannot reach from the start symbol)

C → aCb (cannot derive a terminal string)

can be changed to

S → aS | A

A → a

9

CFG Important Rules

Rule 4. Eliminate the λ production(Rule)

λ production:

A → λ or A =>* λ

Given a CFG G=(V,T,S,P) may generate a language not containing λ, the λ production can be removed

10

Example

G=({S,A},{a,b}, S, P), where P are given by

$S \rightarrow aAb$

$A \rightarrow aAb \mid \lambda$

can be changed to

$S \rightarrow aAb \mid ab$

$A \rightarrow aAb \mid ab$

Can generate the language $L=\{a^n b^n : n >= 1\}$

11

Example

G=({S,A,B,C,D},{a,b,d}, S, P), where P are given by

$S \rightarrow ABaC$

$A \rightarrow BC$

$B \rightarrow b \mid \lambda$

$C \rightarrow D \mid \lambda$

$D \rightarrow d$

can be changed to

$S \rightarrow ABaC \mid BaC \mid AaC \mid ABa \mid aC \mid Aa \mid Ba \mid a$

$A \rightarrow B \mid C \mid BC$

$B \rightarrow b$

$C \rightarrow d$

12

CFG Important Rules

Rule 5. Eliminate the Unit production(Rule)

Unit production:

$A \rightarrow B$, where $A,B \in V$

Given a CFG G=(V,T,S,P) may generate a language not containing unit production, the unit production can be removed

13

Example

G=({S,A,B},{a,b}, S, P), where P are given by

$S \rightarrow Aa \mid B$

$A \rightarrow a \mid bc \mid B$ ($A \rightarrow B$)

$B \rightarrow A \mid bb$ ($B \rightarrow A$)

can be changed to

$S \rightarrow Aa \mid bb$

$A \rightarrow a \mid bc \mid bb$

14

CFG Important Rules usage

The use sequence of the CFG reduction should be as followed:

Rule 1. Substitution and elimination
Rule 2. Eliminate the Left Recursion
Rule 4. Eliminate the λ production (Rule)
Rule 5. Eliminate the Unit production (Rule)
Rule 3. Eliminate the useless production (Rule)

15

Example

$E \rightarrow E + T$

$E \rightarrow T$

Can be changed to

$E \rightarrow T \mid TZ$

$Z \rightarrow +T \mid +TZ$

16

How to do it?

- <term> → <term> (* | /) <factor> | <factor>
- ??
- <term> → <factor> { (* | /) <factor> }

17

Answer

<term> → <term> (* | /) <factor> | <factor>
By rule 2. Eliminate left-recursive
<term> → <factor> | <factor> <Z>
<Z> → (* | /) <factor> | (* | /) <factor><Z>
By EBNF definition
<Z> → (* | /) <factor> { <Z> }
<Z> → (* | /) <factor> { (* | /) <factor> }
<Z> → { (* | /) <factor> }
By rule 5. <Z> is unit production
<term> → <factor> { <Z> }
<term> → <factor> { (* | /) <factor> }

18