2023

Theory of Computation

Kun-Ta Chuang
Department of Computer Science and Information Engineering
National Cheng Kung University

Outline

1 Methods for Transforming Grammars

2 Two Important Normal Form

3 A Membership Algorithm for CFGs*

Theorem 6.1

Let G = (V, T, S, P) be a context-free grammar. Suppose that P contains a production of the form

$$A \rightarrow x_1 B x_2$$

Assume that A and B are different variables and that

$$B \rightarrow y_1 | y_2 | \ldots | y_n$$

is the set of all productions in P which have B as the left side.

Let Ĝ=(V, T, S, Ṗ) be the grammar in which Ṗ is constructed by deleting

$$A \rightarrow x_1 B x_2$$

把B换掉

from P, and adding to it

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

Then

$$L(\hat{G}) = L(G)$$

Example 6.1

Consider G with following productions

$$A \to a \mid aaA \mid abBc$$

$$B \to abbA \mid b$$

各别代入

Using the suggested substitution for the variable B, we get the grammar Ĝ

$$A \to a \mid aaA \mid ab\boxed{abbA}c \mid ab\boxed{b}c$$

Useful Substitution Rules

- **Rule 1:** Remove Nullable Variables ⇒ 移除λ
- **Rule 2:** Remove Unit-Productions ⇒ 移除 $A \rightarrow B$ $B \rightarrow A$
- **Rule 3:** Remove Useless Variables ⇒ 移除没用到的variable

Nullable Variables

$\lambda-\text{production}:$ $A \rightarrow \lambda$

Nullable Variable: $A \Rightarrow \ldots \Rightarrow \lambda$

↳ A有機會變λ

Example 6.4

$$\{a^n b^n : n \geq 1\}$$

S_1用aS_1b和λ代入

$$S \to aS_1b \qquad\qquad S \to aS_1b \mid ab$$

$$S_1 \to aS_1b \mid \lambda \qquad\qquad S_1 \to aS_1b \mid ab$$

以S_1原本的以S_1用λ代入

Example 6.5

Find a CFG without λ-productions equivalent to the grammar G:

Grammar G

$S \rightarrow ABaC$

$A \rightarrow BC$

$B \rightarrow b \mid \lambda$

$C \rightarrow D \mid \lambda$

$D \rightarrow d$

⇨

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

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

$B \rightarrow b$

$C \rightarrow D$

$D \rightarrow d$

A,B, and C are nullable variables

Unit-Productions

Unit Production: $A \rightarrow B$

(a single variable in both sides)

Removing Unit Productions

Observation:

$$A \to A$$ ⇒ 沒用

Is removed immediately

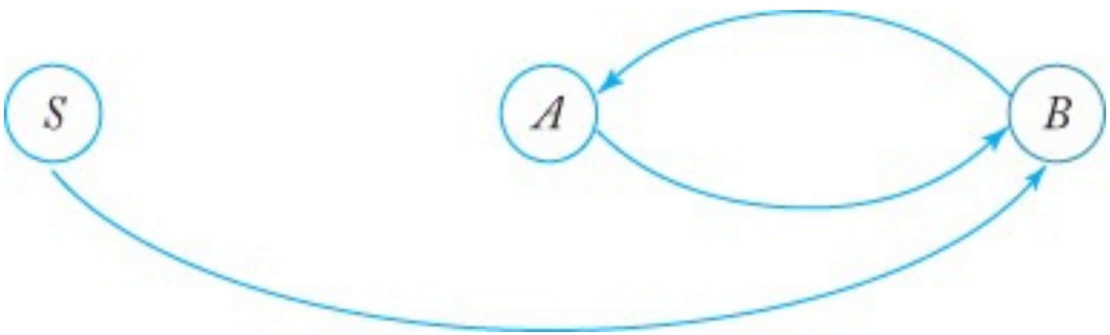
Example 6.6

Remove all unit-productions from

$$S \to Aa \mid B$$

$$B \to \underline{A} \mid bb$$

$$A \to a \mid bc \mid \underline{B}$$



dependency graph

Example 6.6

$$S \to Aa \mid B$$

$$B \to A \mid bb$$

$$A \to a \mid bc \mid B$$

B代入

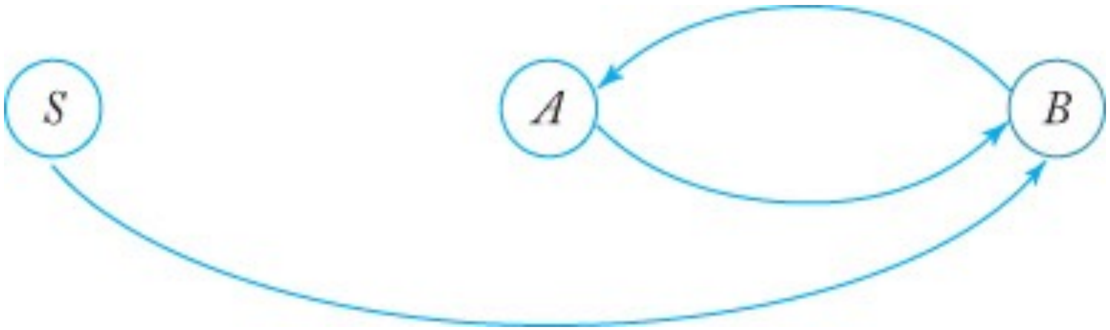
A代入

$$\boxed{\begin{aligned} S &\to Aa \\ B &\to bb \\ A &\to a \mid bc \end{aligned}} + \boxed{\begin{aligned} S &\to a \mid bc \mid bb \\ B &\to a \mid bc \\ A &\to bb \end{aligned}} = \boxed{\begin{aligned} S &\to a \mid bc \mid bb \mid Aa \\ &\cancel{B \to a \mid bb \mid bc} \\ A &\to a \mid bb \mid bc \end{aligned}}$$

Non-unit production

New rules

B用不到



dependency graph

Useless Productions

$$S \to aSb$$

$$S \to \lambda$$

$$S \to A$$

$$A \to aA$$

Useless Production

Some derivations never terminate... 永遠不會停

$$S \Rightarrow A \Rightarrow aA \Rightarrow aaA \Rightarrow \ldots \Rightarrow aa\ldots aA \Rightarrow \ldots$$

Another grammar:

$$S \rightarrow A$$

$$A \rightarrow aA$$

$$A \rightarrow \lambda$$

$$B \rightarrow bA$$

Useless Production

Not reachable from S!

In general:

contains only terminals

if $S \Rightarrow \ldots \Rightarrow xAy \Rightarrow \ldots \Rightarrow w$

有用到

$w \in L(G)$

then variable A is useful

otherwise, variable A is useless

A production $A \to x$ is useless
iff any of its variables is useless

$$S \to aSb$$

$$S \to \lambda$$

Productions

Variables

$S \to A$ — useless

∵ A 為 useless
∴ S → A 也為 useless

useless — $A \to aA$ — useless

useless — $B \to C$ — useless

useless — $C \to D$ — useless

Removing Useless Productions

Example 6.3:

Eliminate useless symbols and productions from the grammar below:

$$S \rightarrow aS \mid A \mid C$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

First:

find all variables that can produce strings with only terminals

↳ 留下能產生 terminal

$$S \to aS \mid A \mid C$$

$$A \to a$$

$$B \to aa$$

$$C \to aCb$$

$$\{A, B\}$$

$$\because S \to A$$

$$\{A, B, S\}$$

Keep only the variables
that produce terminal symbols: $\{A, B, S\}$

(the rest variables are useless)

$$S \rightarrow aS \mid A \mid \cancel{C}$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$\cancel{C \rightarrow aCb}$$

$\Rightarrow$

$$S \rightarrow aS \mid A$$

$$A \rightarrow a$$

$$B \rightarrow aa$$

Remove useless productions

Second: Find all variables reachable from S

Dependency graph

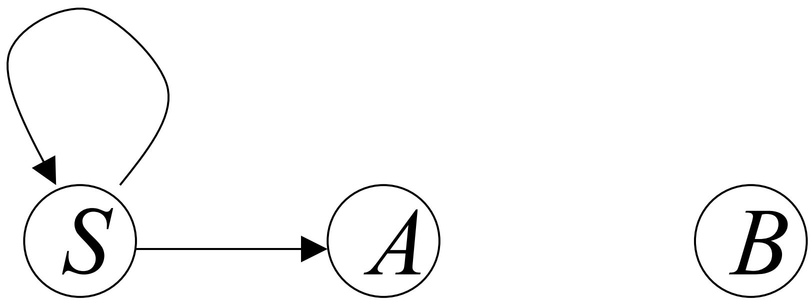
- Vertex labeled with variable
- Edge (A, B) exists iff a production form

A → xBy

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$



not
reachable

S 走不到 B

Keep only the variables
reachable from S

(the rest variables are useless)

Final Grammar

$$S \to aS \mid A$$

$$A \to a$$

$$\cancel{B \to aa}$$

$$S \to aS \mid A$$

$$A \to a$$

Remove useless productions

Theorem 6.5

context-free-language

- Let L be a CFL that does not contain λ. Then there exists a CFG that generates L and that does not have any useless-, unit-, or λ-production.

$$\boxed{S_0 \rightarrow S \mid \lambda}$$

- Which one needs to be removed first?

- Remove all undesirable productions using the following sequence of steps:
- 先 **Step 1:** Remove λ-productions
- **Step 2:** Remove unit-productions
- **Step 3:** Remove useless-productions

Outline

1. Methods for Transforming Grammars
2. Two Important Normal Form
3. A Membership Algorithm for CFGs*

Chomsky Normal Form (CNF)

Each productions has form:

只能有2 variable or 1 terminal

$$A \rightarrow BC \qquad \text{or} \qquad A \rightarrow a$$

variable variable terminal

Noam Chomsky

- The Grammar Guy
- 1928 –
- b. Philadelphia, PA
- PhD – UPenn (1955)
  - Linguistics
- Prof at MIT (Linguistics) (1955 - present)

Example 6.7

$$S \to AS$$
$$S \to a$$
$$A \to SA$$
$$A \to b$$

Chomsky
Normal Form

$$S \to AS$$
$$S \to AAS$$ 3 variable
$$A \to SA$$
$$A \to aa$$ 2 terminal

Not Chomsky
Normal Form

Example 6.8

- Convert the grammar with following productions to CNF:

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

Introduce variables for terminals: T_a, T_b, T_c

$$S \to ABa$$
$$A \to aab$$
$$B \to Ac$$

↳先用 T_a 取代 a
並在最後補上 $T_a \to a$

$$S \to ABT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Introduce intermediate variable: V_1

$$S \to ABT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

拆解成2variable

$$\boxed{\begin{aligned} S &\to AV_1 \\ V_1 &\to BT_a \end{aligned}}$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Introduce intermediate variable: V_2

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$\boxed{\begin{array}{l} A \to T_aV_2 \\ V_2 \to T_aT_b \end{array}}$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Final grammar in Chomsky Normal Form:

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_a V_2$$

$$V_2 \to T_a T_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

Initial grammar

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

Theorem 6.6

From any context-free grammar
(which doesn’t produce λ)
not in Chomsky Normal Form

we can obtain:
An equivalent grammar
in Chomsky Normal Form

The Procedure

First remove:

Nullable variables

Unit productions

Then, for every symbol a :

Add production $T_a \to a$

In productions: replace a with T_a

New variable: T_a

Replace any production $A \to C_1 C_2 \cdots C_n$

with

$$A \to C_1 V_1$$

$$V_1 \to C_2 V_2$$

$$\cdots$$

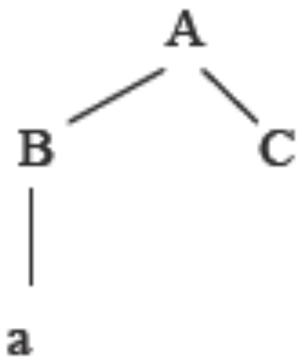
$$V_{n-2} \to C_{n-1} C_n$$

New intermediate variables: $V_1, V_2, \ldots, V_{n-2}$

Theorem: For any context-free grammar (which doesn’t produce λ) there is an equivalent grammar in Chomsky Normal Form

Observations

- Chomsky normal forms are good for parsing and proving theorems



At most 2 branches at every node

↳ binary tree

- It is very easy to find the Chomsky normal form for any context-free grammar

Greibach Normal Form 参考

All productions have form:

Sheila Greibach

PhD (1963) Harvard University

Prof. of UCLA(CS)

$$A \to \underbrace{a}_{\text{symbol}} \, \underbrace{V_1 V_2 \cdots V_k}_{\text{variables}} \qquad k \geq 0$$

Examples:

$$S \to cAB$$

$$A \to aA \mid bB \mid b$$

$$B \to b$$

Greibach
Normal Form

$$S \to abSb$$

$$S \to aa$$

Not Greibach
Normal Form

Example 6.9:

$$
\begin{aligned}
S &\to AB \\
A &\to aA \mid bB \mid b \\
B &\to b
\end{aligned}
\quad \Longrightarrow \quad
\begin{aligned}
S &\to aAB \mid bBB \mid bB \\
A &\to aA \mid bB \mid b \\
B &\to b
\end{aligned}
$$

Example 6.10:

$$S \to abSb$$
$$S \to aa$$

$$S \to aT_bST_b$$
$$S \to aT_a$$
$$T_a \to a$$
$$T_b \to b$$

Greibach
Normal Form

Theorem 6.7:

For any context-free grammar (which doesn't produce λ) there is an equivalent grammar in Greibach Normal Form

Observations

- Greibach normal forms are very good for parsing

- It is hard to find the Greibach normal form of any context-free grammar

口不好用

Outline

1. Methods for Transforming Grammars
2. Two Important Normal Form
3. A Membership Algorithm for CFGs*

Membership Question:

for context-free grammar G
find if string $w \in L(G)$

Membership Algorithms: Parsers

- Exhaustive search parser: $O(P^{2|w|+1})$

暴力解

- **CYK** parsing algorithm: $O(|w|^3)$

11/22

The CYK Parser

J. Cocke
D. H. Younger
T. Kasami

The CYK Membership Algorithm

Input:

- Grammar G in Chomsky Normal Form
  ↳ 先轉成CNF
- String w

Output:

find if $w \in L(G)$

The Algorithm

Input example:

- Grammar G :
$$S \to AB$$
$$A \to BB$$
$$A \to a$$
$$B \to AB$$
$$B \to b$$

- String w : $aabbb$

1 2 3 4 5

$aabbb \ (V_{15})$ → 列出所有組合(按順序)

length	1	2	3	4	5	start position
1	a	a	b	b	b	
2	aa	ab	bb	bb		
3	aab	abb	bbb			
4	aabb	abbb				
5	aabbb					

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$\underline{A \rightarrow a}$$

$$B \rightarrow AB$$

$$\underline{B \rightarrow b}$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
aab	abb	bbb		
aabb	abbb			
aabbb				

↳長度為1，直接對grammer

$S \to AB$

$A \to BB$

$A \to a$

$B \to AB$

$B \to b$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
aabb	abbb			
aabbb				

b, b
↓
BB ⇒ A可生成

拆開
a, a
↓
由 AA 產生
但grammar找不到
能生出AA的

拆開
a, b
↓
A B
⇒ S, B 可生成AB

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
S,B	A	S,B		
aabb	abbb			
A	S,B			
aabbb				
S,B				

✓ ✗

① ②

① a, ab
↓ ↓
A S,B
↳找能生成AS or AB的
⇒ S,B

② aa, b
↓
✗
↳stop

① a, bb
↓ ↓
A A
↳找不到能生成AA的

② ab, b
↓ ↓
S,B B
↳找能生成SB or BB
⇒ A

↳一個string可由grammer生成，則必有S

Therefore: $$aabbb \in L(G)$$

Time Complexity: $|w|^3$

↳ $\underbrace{O(w^2)}_{w\text{為長度}} \times \underbrace{O(w)}_{\text{逗點位置}} = w^3$

$\frac{w^2}{2}$ 每個檢查(w-1)次

Observation: The CYK algorithm can be easily converted to a parser (bottom up parser)