การทำให้เรียบง่าย

Simplifications of Context-Free Grammars

A Substitution Rule

ใช้ กฎการแทนที่

* อยากแทนที่ B

aB

$$S \to aB$$
$$A \to aaA$$
$$A \to abBc$$
$$B \to aA$$

$abBc$

$$B \to b$$

ถ้าอยากกำจัด B ก็แทนด้วย b

Substitute

$$B \to b$$

Equivalent grammar

แทน B ด้วย b

$$S \to aB \mid ab$$
$$A \to aaA$$
$$A \to abBc \mid abbc$$
$$B \to aA$$

A Substitution Rule

$$S \to aB \mid ab$$

$$A \to aaA$$

$$A \to abBc \mid abbc$$

บรรทัด B -> b หายไปแล้ว

$$B \to aA$$

Substitute

$$B \to aA$$

$$S \to \cancel{aB} \mid ab \mid aaA$$

$$A \to aaA$$

$$A \to \cancel{abBc} \mid abbc \mid abaAc$$

Equivalent grammar

ตัดได้ก็ต้องก็อบ B →aA

In general:

ภาษาเหมือนเดิม

Grammar เปลี่ยน

$$A \to xBz$$

$$B \to y_1$$

Substitute

$$B \to y_1$$

$$A \to xBz \mid xy_1z$$

equivalent grammar

Nullable Variables

๑ ขยมกำจัด

ทำให้ sentence ของเกิด loop

มี null

$\lambda -$ production: $\qquad A \rightarrow \lambda$

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

Removing Nullable Variables

Example Grammar:

$$S \to aMb$$

$$M \to aMb$$

$$M \to \lambda$$

Nullable variable

$$S \to aMb$$
$$M \to aMb$$
$$\cancel{M \to \lambda}$$

Substitute
$$M \to \lambda$$

Final Grammar

$$S \to aMb$$
$$S \to ab$$
$$M \to aMb$$
$$M \to ab$$

๒

Unit-Productions

production ที่มี Var การเขียน ไป หา 1 ตัว

Unit Production: $A \rightarrow B$

(a single variable in both sides)

Removing Unit Productions

Observation:

$$\cancel{A \rightarrow A}$$

Is removed immediately

Example Grammar:

$$S \to aA$$

$$A \to a$$

$$\left.\begin{array}{l} A \to B \\ B \to A \end{array}\right\} \text{unit production}$$

$$B \to bb$$

$$S \rightarrow aA$$

$$A \rightarrow a$$

$$\cancel{A \rightarrow B}$$

$$B \rightarrow A$$

$$B \rightarrow bb$$

ใช้อันแรก

Substitute

$$A \rightarrow B$$

$$S \rightarrow aA \mid aB$$

$$A \rightarrow a$$

$$B \rightarrow A \mid B$$

$$B \rightarrow bb$$

$B \rightarrow B$ ตัดทิ้งได้เลย

$$S \rightarrow aA \mid aB$$
$$A \rightarrow a$$
$$B \rightarrow A \mid \cancel{B}$$
$$B \rightarrow bb$$

Remove

$$B \rightarrow B$$

$$S \rightarrow aA \mid aB$$
$$A \rightarrow a$$
$$B \rightarrow A$$
$$B \rightarrow bb$$

$$S \rightarrow aA \mid aB$$

$$A \rightarrow a$$

$$\cancel{B \rightarrow A}$$

$$B \rightarrow bb$$

Substitute

$$B \rightarrow A$$

เหมือนกันตัดทิ้งได้เลย

$$S \rightarrow aA \mid aB \mid aA$$

$$A \rightarrow a$$

$$B \rightarrow bb$$

Remove repeated productions

$$S \to aA \mid aB \mid \cancel{aA}$$

$$A \to a$$

$$B \to bb$$

Final grammar

$$S \to aA \mid aB$$

$$A \to a$$

$$B \to bb$$

simplication

3

Useless Productions

production ควรตัดออก ไร้ประโยชน์

$$S \to aSb$$

$$S \to \lambda$$

$$S \to A$$

$$A \to aA$$ Useless Production

Some derivations never terminate...

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

Another grammar:

$$S \rightarrow A$$

$$A \rightarrow aA$$

$$A \rightarrow \lambda$$

ไม่สามารถรวมมาถึง

$$B \rightarrow bA$$ Useless Production

Not reachable from S

In general:

contains only terminals

มีทางเดินมาจาก

สามารถ derive เป็น terminal

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

var A จะ useful ก็ต้องร่วม link จากหัว ไปจบ

$w \in L(G)$

then variable A is useful

otherwise, variable A is useless

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

$$S \to aSb$$

$$S \to \lambda$$

		Productions
Variables	$S \to A$	useless กำจัดทิ้งแม่ง
useless	$A \to aA$	useless ลงเป็น terminal ไม่ได้
useless	$B \to C$	useless ไม่มีคนมาถึง
	derive เป็นไรไม่ได้	
useless	$C \to D$	useless ไม่มีคนมาหาเลย

Removing Useless Productions

Example Grammar:

เริ่ม

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

$$A \rightarrow a$$

$$B \rightarrow aa$$

$$C \rightarrow aCb$$

First: find all variables that can produce strings with only terminals

① หาvariable ที่สามารถแตกออกเป็น terminal strings ได้

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

$$A \rightarrow a$$ ✓

$$B \rightarrow aa$$ ✓

$$C \rightarrow aCb$$

Round 1: $\{A, B\}$

$$S \rightarrow A$$

ที่ generate variable strings ได้

Round 2: $\{A, B, S\}$

② อะไรที่ derive ได้ A หรือ B

มีแค่ S

อะไรไม่อยู่ในนี้ ลบทิ้งแม่ง

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

(the rest variables are useless)

$$S \to aS \mid A \mid \cancel{C}$$
$$A \to a$$
$$B \to aa$$
$$\cancel{C \to aCb}$$

$$\Rightarrow$$

$$S \to aS \mid A$$
$$A \to a$$
$$B \to aa$$

Remove useless productions

Second: Find all variables reachable from S

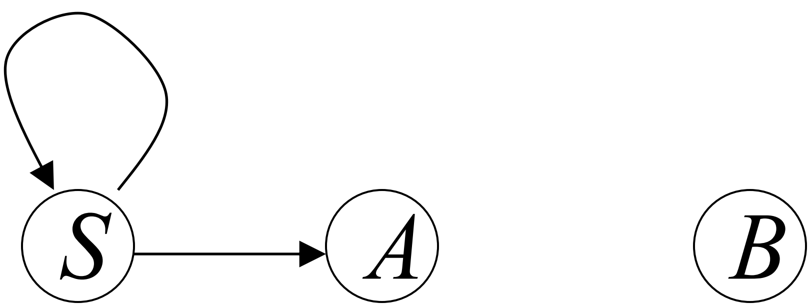
อะไรไปถึง S ได้บ้าง

Use a Dependency Graph

$$S \to aS \mid A$$

$$A \to a$$

$$B \to aa$$



not reachable

Keep only the variables reachable from S

(the rest variables are useless)

$$S \to aS \mid A$$

$$A \to a$$

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

ไปไม่ถึง

Final Grammar

$$S \to aS \mid A$$

$$A \to a$$

Remove useless productions

Removing All

Step 1: Remove Nullable Variables

Step 2: Remove Unit-Productions

Step 3: Remove Useless Variables

ผลอยากปุ่ปนพร้อมใบรงขับ

algorithm parse ต้องการให้ context free อยู่ในรูป normal form ก่อน

Normal Forms
for
Context-free Grammars

แบบที่ 1

Chomsky Normal Form

Each productions has form:

variable 2 ตัวติดกัน

$$A \rightarrow BC$$

variable variable

or

หรือ term

$$A \rightarrow a$$

1 ตัว

terminal

Examples:

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

Chomsky Normal Form

$$S \to AS$$
$$S \to AAS$$
$$A \to SA$$
$$A \to aa$$

Not Chomsky Normal Form

Convertion to Chomsky Normal Form

Example:

$$S \rightarrow ABa$$

$$A \rightarrow aab$$

$$B \rightarrow Ac$$

ไม่ใช่

Not Chomsky
Normal Form

Introduce variables for terminals: T_a, T_b, T_c

วิธการแปลง

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

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

ได้มา 3 ตัวแล้ว

Introduce intermediate variable: V_1

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

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

Introduce intermediate variable: V_2

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

$\Longrightarrow$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aV_2$$
$$V_2 \to T_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Final grammar in Chomsky Normal Form:

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_a V_2$$

$$V_2 \to T_a T_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

ได้ยาวขึ้น

↓

ตามแลดตามง่าย

Initial grammar

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

In general:

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

สำหรับ cfg grammar ที่ไม่ได้อยู่ใน chomsky NF
เราสามารถสร้าง grammar ที่อยู่ใน NF ได้

we can obtain:

An equivalent grammar
in Chomsky Normal Form

The Procedure

First remove:

Nullable variables

Unit productions

Then, for every symbol a:

กรณีแทน terminal ได้ด้วย บนง

Add production $\quad T_a \rightarrow a$

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

รูปแบบที่ดี

- Chomsky normal forms are good for parsing and proving theorems

แปลงง่าย

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

ข

Greibach Normal Form

กรีซึก

All productions have form:

1 ตัว 1 ตัว Var เท่าไหร่ก็ได้

$$A \rightarrow a\underbrace{V_1 V_2 \cdots V_k}_{} \qquad k \geq 0$$

symbol variables

Examples:

$$S \rightarrow cAB$$

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

$$B \rightarrow b$$

Greibach
Normal Form

$$S \rightarrow a\cancel{b}S\cancel{b}$$

$$S \rightarrow a\cancel{a}$$

Not Greibach
Normal Form

Conversion to Greibach Normal Form:

เป็นอีกวิธีที่แปลงได้

$$S \to abSb$$

$$S \to aa$$

ใช้หลักการ

แทน ter → $\overline{T}_{ter}$

$$S \to aT_bST_b$$

$$S \to aT_a$$

$$T_a \to a$$

$$T_b \to b$$

Greibach

Normal Form

Theorem: 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 possible to find the Greibach normal form of any context-free grammar

The CYK Parser
(Cocke-Younger-Kasami)

algorithm ที่ใช้ในการ parse string [illegible]

The CYK Membership Algorithm

how to check if $w \in L$

Input:

- Grammar G in Chomsky Normal Form
- String w

$w \in L(G)$

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 : $aabbb$

aabbb

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

ต้องๆ ขนาด window

การ slide

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

δ=1 sdemelõ →

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

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

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

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

a A	a A	b B	b B	b B
aa $\boxed{}$	ab S,B	bb A	bb A	
aab S,B	abb A	bbb S,B		
aabb A	abbb S,B			
aabbb S,B				

$S \to AB$

$A \to BB$

$A \to a$

$B \to AB$

$B \to b$

เส้นนี้คือดูว่า string abb แตกมาจากอะไรได้บ้าง

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

$$S \rightarrow AB$$

$$A \rightarrow BB$$

$$A \rightarrow a$$

$$B \rightarrow AB$$

$$B \rightarrow b$$

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

$S \to AB$

$A \to BB$

$A \to a$

$B \to AB$

$B \to b$

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

ทำแบบเดียวกันไปเรื่อยๆ

อันท้ายๆ เหมือนเดิม

ถ้าบรรทัดสุดท้ายมี S อยู่ → มันอยู่ใน L(G) นี้

Therefore: $aabbb \in L(G)$

3 for loop

Time Complexity: $|w|^3$

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