

# **Automata**

————— **Final** —————

# **Grammar & Turing Machine**

**version 1**

**17/5/2018**

## Guideline

### Grammar

- Structure
- Right Linear Grammar (RLG)
- Left Linear Grammar (LLG)
- Convert RLG  $\rightarrow$  LLG
  - RLG  $\rightarrow$  NFA
  - NFA  $\rightarrow$  NFA<sup>R</sup>
  - NFA<sup>R</sup>  $\rightarrow$  RLG<sup>R</sup>
  - RLG<sup>R</sup>  $\rightarrow$  LLG
- Context Free Grammar
  - Find CFG from L
- Normalization
- Chomsky's Normal Form
- Parsing String
  - Bottom-up : CYK
  - Top-down : Earley's Method

### Nondeterministic Pushdown Automata (NPDA)

- Structure
- Convert CFG  $\rightarrow$  NPDA
- Convert NPDA  $\rightarrow$  CFG

### Turing Machine

- Structure
- Type
- Halt

## Content

<b>Guideline</b>	<b>2</b>
<b>Content</b>	<b>3</b>
<b>Grammar</b>	<b>4</b>
Structure	4
Right Linear Grammar (RLG)	5
Left Linear Grammar (LLG)	5
Convert RLG $\rightarrow$ LLG	5
Convert RLG $\rightarrow$ NFA	5
Convert NFA $\rightarrow$ NFAR	5
Convert NFAR $\rightarrow$ RLGR	6
Convert RLGR $\rightarrow$ LLG	6
Example Convert RLG $\rightarrow$ LLG	7
<b>Normalization</b>	<b>10</b>
Step 1 Remove Unreachable and useless production	10
Step 2 Remove $\lambda$ production	12
Step 3 Remove unit production	12
<b>Chomsky's Normal Form</b>	<b>13</b>

# Grammar

## Structure

### Definition 1

Grammar คือการเขียนอธิบาย syntax ของภาษาใดๆ โดยแต่ละ grammar จะประกอบด้วย 4 อย่าง ได้แก่

- Set of Variables (V)
- Set of Terminal (T)
- Starting Variable (S)
- Set of Productions (P)

$$G = (V, T, S, P)$$

### Definition 2

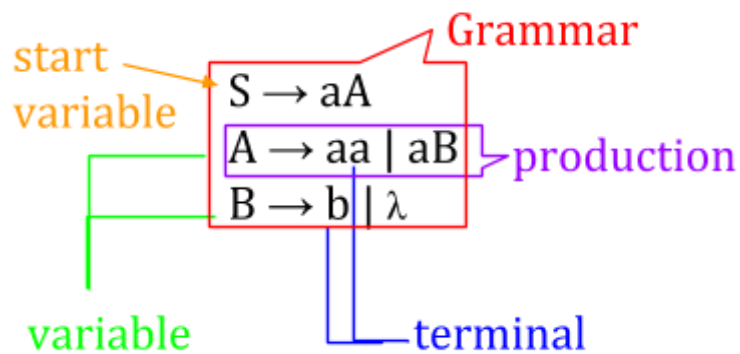
Terminal คือ อักษรต่างๆ ที่ภาษานั้นๆสามารถรับได้

### Definition 3

Variable คือ กลุ่มของอักขระ

### Definition 4

Production คือ การเขียนอธิบาย Variable ต่างๆว่าเกิดจากอะไร



## Right Linear Grammar (RLG)

ทุกๆ production มีสมบัติดังนี้

- มี variable เพียงตัวเดียวและอยู่ขวาสุด
- มี terminal ที่ตัวก็ได้

$$A \rightarrow x^*B$$

## Left Linear Grammar (LLG)

ทุกๆ production มีสมบัติดังนี้

- มี variable เพียงตัวเดียวและอยู่ซ้ายสุด
- มี terminal ที่ตัวก็ได้

$$A \rightarrow Bx^*$$

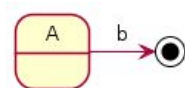
## Convert RLG $\rightarrow$ LLG

$$\text{RLG} \rightarrow \text{NFA} \rightarrow \text{NFA}^R \rightarrow \text{RLG}^R \rightarrow \text{LLG}$$

## Convert RLG $\rightarrow$ NFA

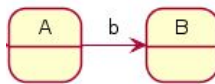
1.  $A \rightarrow b$

$\Rightarrow$



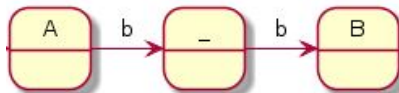
2.  $A \rightarrow bB$

$\Rightarrow$



2.1.  $A \rightarrow bbB$

$\Rightarrow$



// state กลางไม่ต้องมี \_ ก็ได้นะ

3.  $A \rightarrow \lambda$

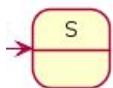
$\Rightarrow$



// คือโปรแกรมมันไม่มีวงกลม 2 ชั้น ใช้ final แทนนะ

4.  $S$


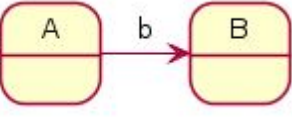
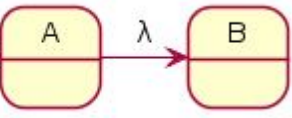
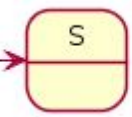
$\Rightarrow$



## Convert NFA $\rightarrow$ NFAR

1. กลับด้าน transition ทั้งหมด
2. เพิ่ม  $S'_0$  เป็น start state ใหม่
3. เพิ่ม transition  $\lambda$  จาก  $S'_0$  ไปยัง final state เก่า
4. เปลี่ยน final state เก่าเป็น normal state
5. เปลี่ยน starting state เก่าเป็น final state

### Convert NFAR $\rightarrow$ RLGR

1.   $\Rightarrow A \rightarrow \lambda$
2.   $\Rightarrow A \rightarrow bB$
3.   $\Rightarrow A \rightarrow B$
4.   $\Rightarrow S$

### Convert RLGR $\rightarrow$ LLG

1.  $A \rightarrow b \Rightarrow A \rightarrow b$
2.  $A \rightarrow bB \Rightarrow A \rightarrow Bb$
- 2.1.  $A \rightarrow abB \Rightarrow A \rightarrow Bba$

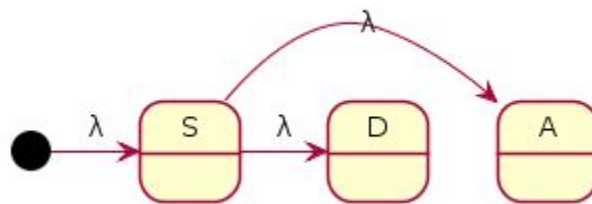
## Example Convert RLG $\rightarrow$ LLG

$S \rightarrow A \mid D$   
 $A \rightarrow aB \mid \lambda$   
 $B \rightarrow bC$   
 $C \rightarrow A$   
 $D \rightarrow bE \mid \lambda$   
 $E \rightarrow aS \mid aF$   
 $F \rightarrow D$

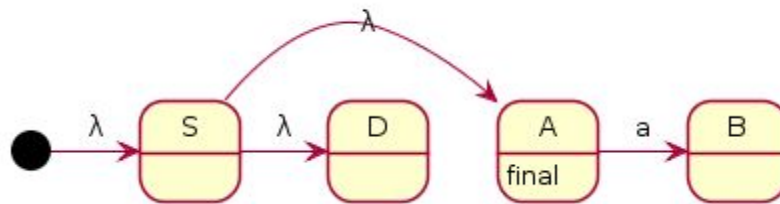
### Step 1 convert RLG to NFA

#### Step 1.1 $S \rightarrow A \mid D$

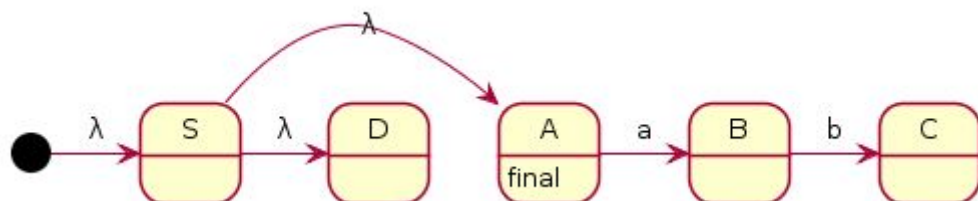
// คือโปรแกรมมันมี state ดำตอนเริ่มมาให้ ชก.ลบแล้ว เอาเป็นว่าเข้าใจนะว่าไม่ต้องมีก็ได้



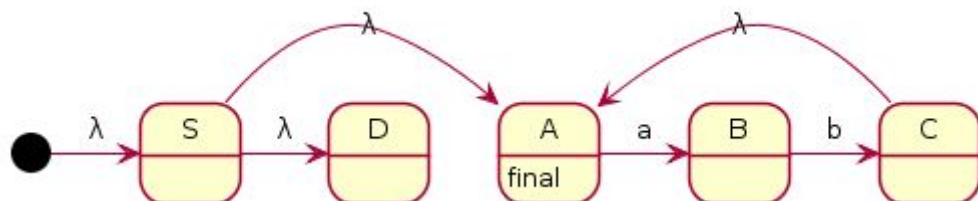
#### Step 1.2 $A \rightarrow aB \mid \lambda$



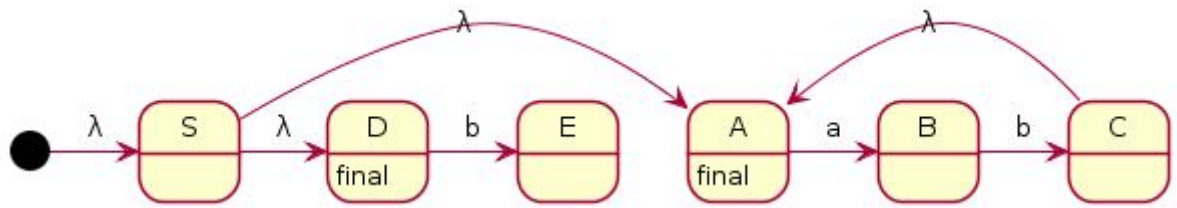
#### Step 1.3 $B \rightarrow bC$



#### Step 1.4 $C \rightarrow A$

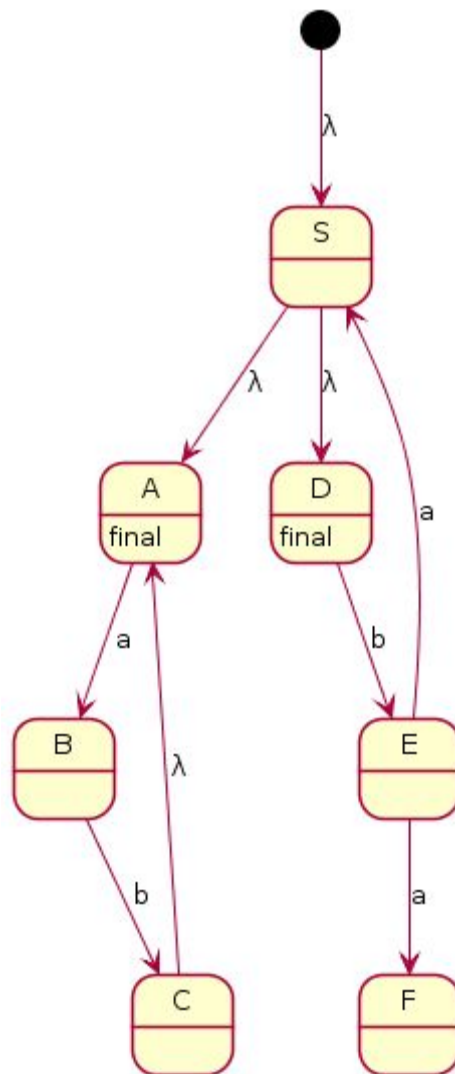


**Step 1.5**      $D \rightarrow bE \mid \lambda$



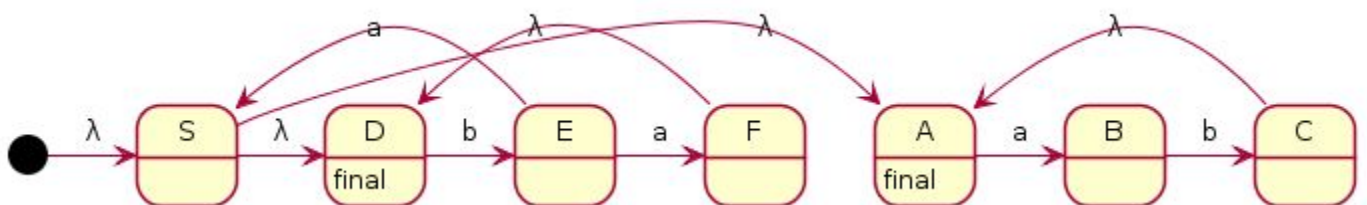
**Step 1.6**      $E \rightarrow aS \mid aF$

// อยู่ดีๆก็เปลี่ยนเป็นแนวตั้ง 5555



**Step 1.7**      $F \rightarrow D$

// แล้วก็กลับมาเป็นแนวนอนเอง

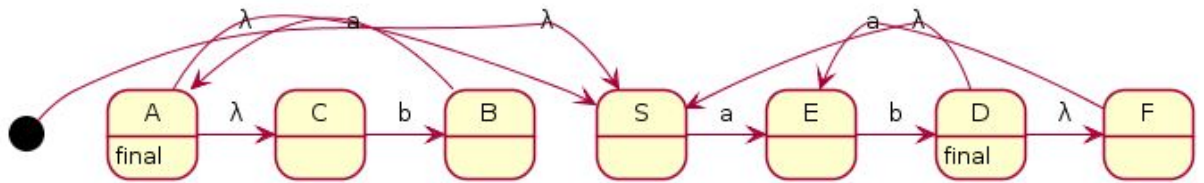




## Step 2 Convert NFA $\rightarrow$ NFA<sup>R</sup>

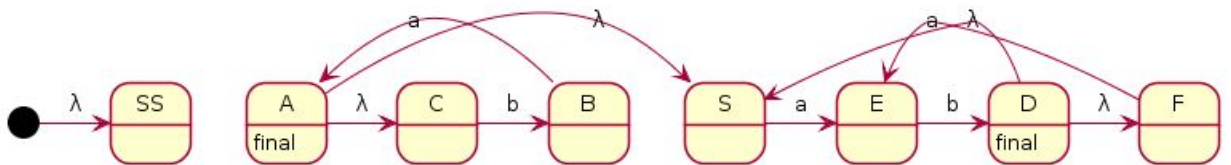
### 1. Reverse all transition

// เส้นดุงงๆ เอาเป็นว่าเข้าใจละกัน 555

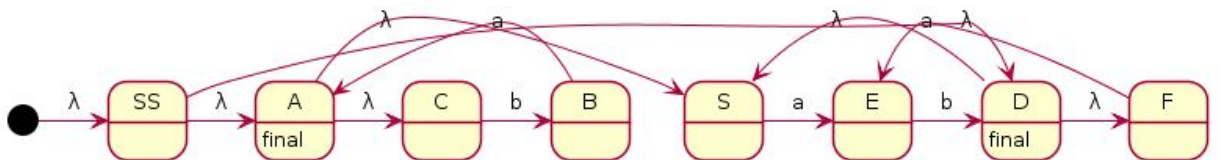


### 2. Add S' to new start state

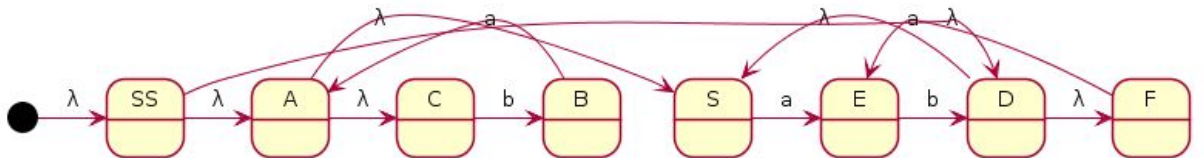
// ใช้ S' ไม่ได้ ขอใช้ SS แทนละกัน



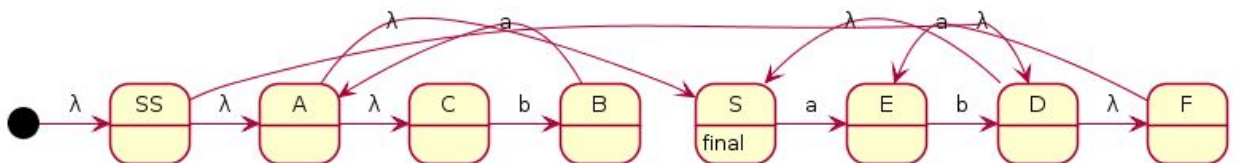
### 3. Add $\lambda$ transition from S' to final state



### 4. Change Final state to normal state



### 5. Change old start state to final state



## Step 3 Convert NFA<sup>R</sup> $\rightarrow$ RLG<sup>R</sup>

$S' \rightarrow A \mid D$   
 $S \rightarrow aE \mid \lambda$   
 $A \rightarrow C \mid S$   
 $B \rightarrow aA$   
 $C \rightarrow bB$   
 $D \rightarrow S \mid F$   
 $E \rightarrow bD$   
 $F \rightarrow aE$

## Step 4 Convert RLG<sup>R</sup> $\rightarrow$ LLG

$S' \rightarrow A \mid D$   
 $S \rightarrow Ea \mid \lambda$   
 $A \rightarrow C \mid S$   
 $B \rightarrow Aa$   
 $C \rightarrow Bb$   
 $D \rightarrow S \mid F$   
 $E \rightarrow Db$   
 $F \rightarrow Ea$

## Normalization

ทำเพื่อปรับปรุง grammar ให้ดีขึ้น แบ่งออกเป็น 3 step

1. Remove unreachable and useless production
2. Remove  $\lambda$  production
3. Remove unit production

### Example

$S \rightarrow aSb \mid A$   
 $A \rightarrow B \mid Caa \mid BAc$   
 $B \rightarrow \lambda \mid BaBc$   
 $C \rightarrow DaC \mid CbD$   
 $D \rightarrow CbD \mid DaC$   
 $E \rightarrow aFCb \mid FaCb$   
 $F \rightarrow a \mid \lambda$

## Step 1 Remove Unreachable and useless production

### Method 1 Teacher Style

1. Remove unreachable

หา production ที่ไม่มีใครเข้าถึงแล้วลบออก

E, F เป็น unreachable ตัดออกไปเลย ไม่ต้องสนใจมันแล้ว

2. สร้างเซต  $V'$  ขึ้นมาเอา Terminal ทั้งหมดใส่ไป (รวม  $\lambda$  ด้วย)

$V' = \{a, b, c, \lambda\}$

3. ใส่ดูทุก production ถ้า **ทุกตัว** ใน production อยู่ใน  $V'$  ให้เพิ่ม variable นั้นเข้า  $V'$

$S \rightarrow aSb$  มีแค่ a, b อยู่ใน  $V'$  ไม่เอา

$S \rightarrow A$  ไม่มีตัวไหนอยู่ใน  $V'$  เลย ไม่เอา

...

$B \rightarrow \lambda$  มี  $\lambda$  อยู่ใน  $V'$   $\Rightarrow V' = \{a, b, c, \lambda, B\}$

ครบทุกตัวแล้ววนกลับไปใส่ใหม่

$A \rightarrow B$  มี B อยู่ใน  $V'$   $\Rightarrow V' = \{a, b, c, \lambda, B, A\}$

...

ใส่ไปจนจบจะได้  $V' = \{a, b, c, \lambda, B, A, S\}$

4. Production ที่ไม่อยู่ใน  $V'$  คือ useless

C, D เป็น useless

ลบ production ที่มี C, D ออกทั้งหมด

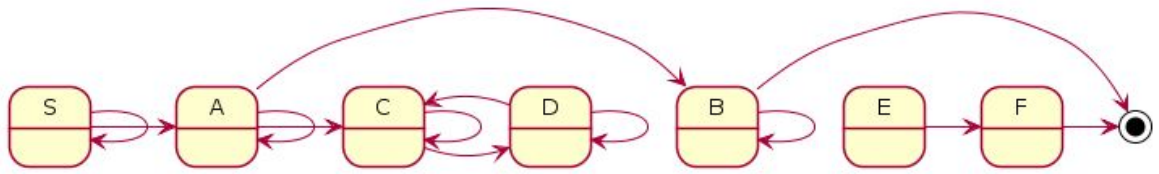
**Result**       $S \rightarrow aSb \mid A$   
                   $A \rightarrow B \mid BAc$   
                   $B \rightarrow \lambda \mid BaBc$

## Method 2 AeAe Style

อันนี้เป็นวิธีที่เรียบเรียงขึ้นเอง เพื่อเอาไปตรวจคำตอบหรืออยากสัๆ

### 1. สร้าง Production Graph

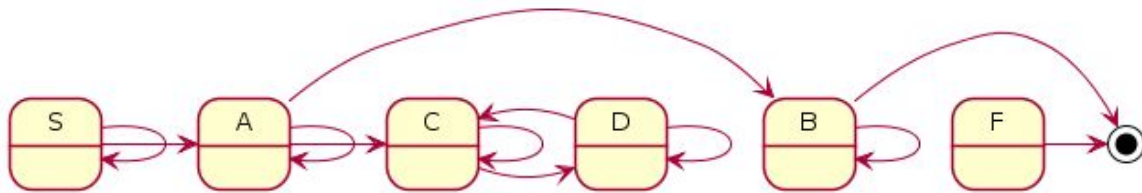
- ถ้ามี Production ไปหา Variable ไหน ก็สร้าง transition ไปหา state Variable นั้น
- ถ้ามี production ที่เป็น terminal **ล้วนๆ** ให้ลากไปหา final state



### 2. Remove unreachable

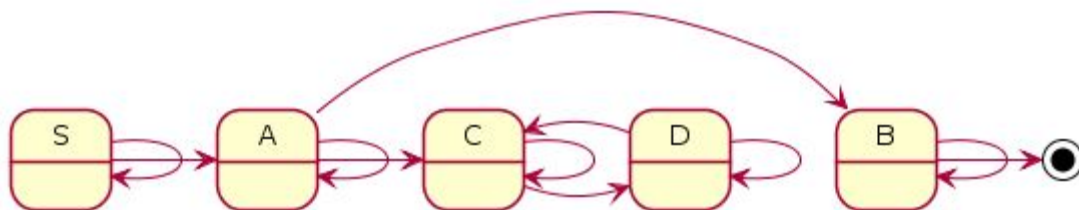
หา state ที่ไม่มีเส้นเข้าเลย ลบออกไป ลบเส้นออกจากมันไปด้วย (ไม่นับ S นะ อย่าลบ S นะเว้ย)

- ลบ E



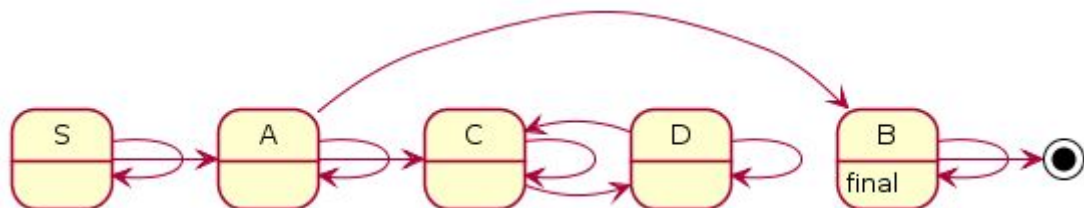
ดูต่อไปเรื่อยๆ จนมีเส้นเข้าทุกอันแล้ว

- ลบ F

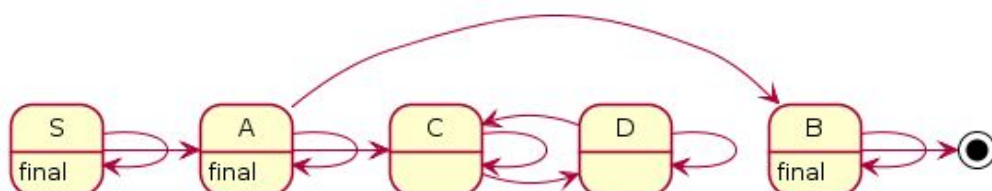
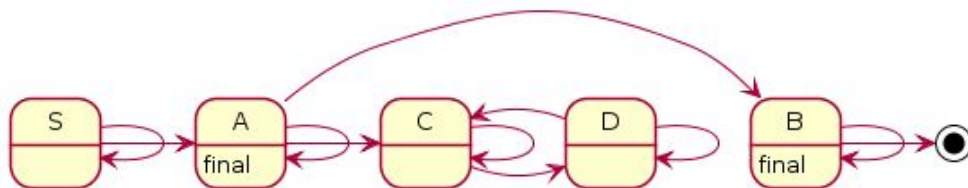


### 3. Remove Useless

- มองหาวามี state ไหนชี้ไปหา final state บ้าง, เปลี่ยนให้เป็น final state เหมือนกัน



- มองต่อไปเรื่อยๆ จนหมดแล้ว (ตอนจบ S ต้องเป็น final แน่แน่นอน)



- State ที่ไม่เป็น final คือ useless

### 4. กลับไปที่ขั้นที่ 1 เช็คให้แน่ใจว่าไม่มี useless และ unreachable

## Step 2 Remove $\lambda$ production

1. หา production ที่มี  $\lambda$   
 $B \rightarrow \lambda$
2. หา production อื่นที่มี variable นี้อยู่  
 $A \rightarrow B$   
 $A \rightarrow BA c$   
 $B \rightarrow BaBc$
3. มอง variable เป็นเหมือน binary ที่มี 0, 1 โดย 0 หมายถึง ไม่ปรากฏ  
 $A \rightarrow B \Rightarrow A \rightarrow \lambda \mid B \quad (0 \mid 1)$   
 $A \rightarrow BA c \Rightarrow A \rightarrow Ac \mid BA c \quad (0 \mid 1)$   
 $B \rightarrow BaBc \Rightarrow B \rightarrow ac \mid aBc \mid Bac \mid BaBc \quad (00 \mid 01 \mid 10 \mid 11)$

**Result**  $S \rightarrow aSb \mid A$   
 $A \rightarrow \lambda \mid B \mid Ac \mid BA c$   
 $B \rightarrow ac \mid aBc \mid Bac \mid BaBc$

4. กลับไปทำขั้น 1 จนไม่เหลือ  $\lambda$  (ยกเว้น Starting Variable มี  $\lambda$  ได้)

$A \rightarrow \lambda$   
 $S \rightarrow A \Rightarrow S \rightarrow \lambda \mid A$   
 $A \rightarrow Ac \Rightarrow A \rightarrow c \mid Ac$   
 $A \rightarrow BA c \Rightarrow A \rightarrow Bc \mid BA c$

**Result**  $S \rightarrow aSb \mid A \mid \lambda$   
 $A \rightarrow B \mid c \mid Ac \mid Bc \mid BA c$   
 $B \rightarrow ac \mid aBc \mid Bac \mid BaBc$

## Step 3 Remove unit production

1. หา production ที่มีแค่ variable ตัวเดียว  
 $A \rightarrow B$
2. ยก production ทั้งหมดของตัวหลังมาแทนที่  
 $A \rightarrow ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BA c$
3. มองหาตัวอื่นและทำให้ครบ

$S \rightarrow A$   
 $S \rightarrow aSb \mid \lambda \mid ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BA c$

**Result**  $S \rightarrow aSb \mid \lambda \mid ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BA c$   
 $A \rightarrow ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BA c$   
 $B \rightarrow ac \mid aBc \mid Bac \mid BaBc$

## Chomsky's Normal Form

เป็นวิธีการปรับให้ Grammar ไม่กำกวม (ambiguous)

Normalize Grammar ก่อนทำ Chomsky

1. เพิ่ม  $S_0 \rightarrow S$

$$S_0 \rightarrow S$$

$$S \rightarrow aSb \mid \lambda \mid ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BAc$$

$$A \rightarrow ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BAc$$

$$B \rightarrow ac \mid aBc \mid Bac \mid BaBc$$

2. Normalize อีกที่

a. หา unreachable, useless

b. Remove  $\lambda$  production

$$S \rightarrow \lambda$$

$$S_0 \rightarrow S \quad \Rightarrow \quad S_0 \rightarrow \lambda \mid S$$

c. Remove unit production

$$S_0 \rightarrow S \quad \Rightarrow \quad S_0 \rightarrow \lambda \mid aSb \mid \lambda \mid ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BAc$$

**Result**

$$S_0 \rightarrow \lambda \mid aSb \mid ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BAc$$

$$S \rightarrow aSb \mid ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BAc$$

$$A \rightarrow ac \mid aBc \mid Bac \mid BaBc \mid c \mid Ac \mid Bc \mid BAc$$

$$B \rightarrow ac \mid aBc \mid Bac \mid BaBc$$

3. แยกทุก production ให้อยู่ในรูป

$$A \rightarrow BC \quad // \text{ มี variable 2 ตัว}$$

$$A \rightarrow a \quad // \text{ มี terminal 1 ตัว}$$

a. สร้าง variable ของแต่ละ terminal ขึ้นมาก่อนจะได้ทำต่อง่ายๆ

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

$$T_c \rightarrow c$$

b. ที่เหลือก็แยกออกเป็นส่วนๆ

$$S_0 \rightarrow \lambda \quad \Rightarrow \quad S_0 \rightarrow \lambda$$

$$S_0 \rightarrow aSb \quad \Rightarrow \quad S_0 \rightarrow T_a R_1$$

$$R_1 \rightarrow S T_b$$

$$S_0 \rightarrow ac \quad \Rightarrow \quad S_0 \rightarrow T_a T_c$$

... ทำไปเรื่อยๆอะ ที่เกียจเขียนแล้ว 555

## CYK

ใช้เพื่อสร้าง Derivation Tree แบบ bottom-up

**Example**

$$S \rightarrow AB$$

$$A \rightarrow aAa \mid bA \mid \lambda$$

$$B \rightarrow aBb \mid \lambda$$

**Example 1**

$$w = aabb$$

**Step 0 ทำ Chomsky's Normal Form**

$$S \rightarrow AB \mid A_1T_a \mid A_2T_b \mid T_aT_a \mid T_bT_b \mid B_1T_b \mid T_aT_b \mid \lambda$$

$$A \rightarrow A_1T_a \mid A_2T_b \mid T_aT_a \mid T_bT_b$$

$$B \rightarrow B_1T_b \mid T_aT_b$$

$$B_1 \rightarrow T_aB$$

$$A_1 \rightarrow T_aA$$

$$A_2 \rightarrow T_bA$$

$$T_a \rightarrow a$$

$$T_b \rightarrow b$$

**Step 1 สร้างบันไดจาก w**

4				
3				
2				
1				
	a	a	b	b

**Step 2 เขียนกำกับแต่ละช่องไว้ว่าคืออะไรจะได้ไม่ลืม**

i	ช่องนี้หมายถึง นับจากตัวที่ k ไป i ตัว
	k

4	aabb			
3	aab	abb		
2	aa	ab	bb	
1	a	a	b	b
	a	a	b	b

### Step 3 มองหา production ที่ทำให้เกิดตัวสีฟ้าได้ โดยดูจากแถวที่ต่ำกว่า

4	aabb			
3	aab	abb		
2	aa	ab	bb	
1	a $T_a$	a $T_a$	b $T_b$	b $T_b$
	a	a	b	b

4	aabb			
3	aab	abb		
2	aa $A, S_0$	ab $S_0, B$	bb $A, S_0$	
1	a $T_a$	a $T_a$	b $T_b$	b $T_b$
	a	a	b	b

$T_b T_b$

$A \rightarrow T_b T_b$   
 $S_0 \rightarrow T_b T_b$

ช่อง abb อาจเกิดจาก (ab)b หรือ a(ab) ก็ได้ ต้องดูทั้ง 2 แบบ

4	aabb			
3	aab $B_1$	abb $A_1$		
2	aa $A, S_0$	ab $S_0, B$	bb $A, S_0$	
1	a $T_a$	a $T_a$	b $T_b$	b $T_b$
	a	a	b	b

$S_0 T_b$   
 $B T_b$

$T_a A$   
 $T_a S_0$

$A_1 \rightarrow T_a A$

<b>4</b>	aabb A, S <sub>0</sub>	ตรงย่อต้องมี start variable ถึงแปลว่ารับ w		
<b>3</b>	aab B <sub>1</sub>	abb A <sub>1</sub>		
<b>2</b>	aa A, S <sub>0</sub>	ab S <sub>0</sub> . B	bb A, S <sub>0</sub>	
<b>1</b>	a T <sub>a</sub>	a T <sub>a</sub>	b T <sub>b</sub>	b T <sub>b</sub>
	<b>a</b>	<b>a</b>	<b>b</b>	<b>b</b>



## Earley's Algorithm

เป็นการสร้าง Rerivation Tree เหมือน CYK แต่จะเป็นแบบ Top-down

\*\* ไม่เน้นใจนะครับ งามมาก

**Example Grammar :**  $S \rightarrow SSS \mid a \mid b$

**Word :**  $w = aba$

**Step 1** เพิ่ม  $S_0$  และ แก้ว  $w$  ตามนี้

$S_0 \rightarrow S-$

$w \rightarrow aba-$

**Step 2** การทำแต่ละรอบจะแบ่งเป็น 3 ขั้นตอน prediction, scan, complete

**Prediction** คือการดู production ว่ามีอะไรบ้าง

**Scan** คือการมองหา production ที่ตรงกับ character ของ  $w$

**Complete** คือการลดรูป production กลับไปเป็น production ก่อนหน้า // ไม่ต้องงง ภู่งง

. แสดงจุดยืนว่ากำลังอยู่ตำแหน่งไหน

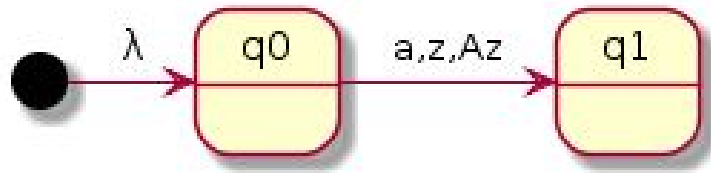
step	state	prediction	origin
<b>Prediction</b> รอบแรก แจกแจงทุก production ว่ามีอะไรบ้าง . อยู่หน้าสุดเพราะแค่นี้มองดู production เลยๆ State กับ origin ถ้าไม่อยากรู้มาก เอาเป็นว่ามันเหมือนกัน 5555			
predict	0	$S_0 \rightarrow .S-$	0
	0	$S \rightarrow .SSS$	0
	0	$S \rightarrow .a$	0
	0	$S \rightarrow .b$	0
<b>Scan</b> หา production ที่หลัง . ตรงกับ ตัวที่ต้องการหา เลื่อน . ไป เพราะตอนนี้ก้าวไปเก็บ a แล้ว			
scan a	1	$S \rightarrow a.$	0
<b>Complete</b> Scan ได้ $S \rightarrow a$ ใช่มั้ย งั้นก็ดูว่ามี production ไหนมี $S$ บ้าง			
Complete	1	$S_0 \rightarrow S.-$	0
	1	$S \rightarrow S.SS$	0
<b>Predict</b> รอบต่อไป เอาแค่ variable ที่อยู่หลัง . ใน complete รอบก่อนหน้ามาทำก็พอ			
Predict	1	$S \rightarrow .SSS$	1
	1	$S \rightarrow .a$	1
	1	$S \rightarrow .b$	1
<b>Scan</b> ทำเหมือนเดิม			
Scan b	2	$S \rightarrow b.$	1

<b>Complete</b> ให้ดึงจาก complete รอบก่อนมาเลื่อน . และทำ complete ใหม่ของรอบตัวเองด้วย			
<b>Complete</b>	2	$S \rightarrow SS.S$	0
	2	$S \rightarrow S.SS$	1
<b>Predict</b> ทำเหมือนเดิม			
<b>Predict</b>	2	$S \rightarrow .SSS$	2
	2	$S \rightarrow .a$	2
	2	$S \rightarrow .b$	2
<b>Scan a</b>	3	$S \rightarrow a.$	2
<b>Complete</b>	3	$S \rightarrow SSS.$	0
	3	$S \rightarrow SS.S$	1
	3	$S \rightarrow S.SS$	2
ถ้า complete แล้ว . อยู่หลังสุด ให้เอามาทำ complete อีกครั้ง			
	3	$S_0 \rightarrow S.- $	0
	3	$S \rightarrow S.SS$	0
เริ่มไม่แน่ใจตอนจบนี้แหละ เหมือนพอถึงตัวสุดท้ายแล้วเอา complete มาดูเลยมั้ง ถ้า origin เป็น 0 ก็แสดงว่ารับ			
<b>Scan - </b>	4	$S_0 \rightarrow S- .$	0

เอาเป็นว่างมากๆ

## Nondeterministic Pushdown Automaton (NPDA)

เป็น automaton แบบที่ทำงานกับ stack โดยแต่ละ transition จะมี 3 ค่า



จากรูปหมายความว่า จะย้ายจาก  $q_0$  ไป  $q_1$  เมื่อมี **a** เป็น input เข้ามา และ top stack = z  
เมื่อตรงกับเงื่อนไขแล้วจะทำการ pop z ออกจาก stack  
หลังจากย้าย state แล้วจะทำการ push Az กลับลงไปใน stack (top stack = A)

### การเขียนอธิบาย transition

$$\delta(q, a, z) = (p, x)$$

a = alphabet ที่เข้ามา

q = state ปัจจุบัน

z = top stack

p = next state

x = push ลง stack

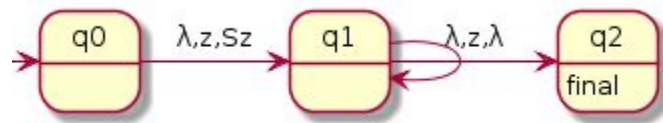
ถ้า  $x = \lambda$  หมายความว่าไม่ push อะไรกลับลง stack เลย

### การ accept string

- Stack ต้องหมดพร้อมกับ string

## Convert CFG $\rightarrow$ PDA

Step 1 สร้าง PDA ตามนี้ไว้ก่อนเลย



Step 2 แปลง Production ต่างๆเป็น transition ดังนี้

$$A \rightarrow x_i \dots x_j \Rightarrow \lambda, A, x_i \dots x_j$$

Step 3 แปลง terminal ทั้งหมดเป็น transition ดังนี้

$$a \Rightarrow a, a, \lambda$$

Step 4 เอา transition ทั้งหมด ใส่ไว้ที่  $q_1 \rightarrow q_1$

## Example

$$S \rightarrow SS \mid aA$$

$$A \rightarrow bAb \mid \lambda$$

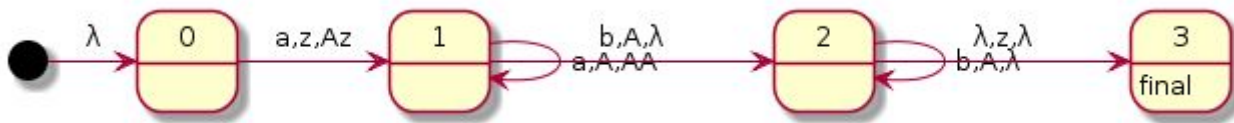
## Result



เออ รูปมันเหี้ย อย่าสนใจเลย

## Convert PDA → CFG

### Example



#### Step 1 สร้าง production S

$S \rightarrow 0zi$  โดยที่  $i$  เป็น state ทุก state

Result  $S \rightarrow 0z0 \mid 0z1 \mid 0z2 \mid 0z3$

#### Step 2 ทา transition ที่ push $\lambda$ มาสร้าง production

(start pop stop)  $\rightarrow$  input

Result  $(1A2) \rightarrow b$  // มาจากเส้น  $1 \rightarrow 2$

$(2A2) \rightarrow b$  // มาจากเส้น  $2 \rightarrow 2$

$(2z3) \rightarrow \lambda$  // มาจากเส้น  $2 \rightarrow 3$

#### Step 3 สร้าง production จาก transition ที่เหลือ

Transition  $\delta(0, a, z) = (1, Az)$  // หมายความว่ามาจากไหนไปดูหน้าก่อนๆ

##### Step 3-1 สร้าง production ตั้งแต่ต้นรอไว้ (มาจากไหนดูตามสีเอานะ)

ตัวหลังสุดใส่ทุก state

$(0z0) \rightarrow$

$(0z1) \rightarrow$

$(0z2) \rightarrow$

$(0z3) \rightarrow$

##### Step 3-2 แยก push ออกเป็นวงเล็บๆ

// จำนวน production ของแต่ละอันเท่ากับจำนวน state ที่มี

$(0z0) \rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$(0z1) \rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$(0z2) \rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$(0z3) \rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

$\rightarrow a(\_ A\_)(\_ z\_)$

### Step 3-3 กระจาย stop state ไปไว้ข้างหน้าสุด

(0z0)  $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
(0z1)  $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
(0z2)  $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
(0z3)  $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$   
 $\rightarrow a(1 A \_)(\_ z \_)$

### Step 3-4 หลังสุดได้ state เหมือนข้างหน้า

(0z0)  $\rightarrow a(1 A \_)(\_ z 0)$   
 $\rightarrow a(1 A \_)(\_ z 0)$   
 $\rightarrow a(1 A \_)(\_ z 0)$   
 $\rightarrow a(1 A \_)(\_ z 0)$   
(0z1)  $\rightarrow a(1 A \_)(\_ z 1)$   
 $\rightarrow a(1 A \_)(\_ z 1)$   
 $\rightarrow a(1 A \_)(\_ z 1)$   
 $\rightarrow a(1 A \_)(\_ z 1)$   
(0z2)  $\rightarrow a(1 A \_)(\_ z 2)$   
 $\rightarrow a(1 A \_)(\_ z 2)$   
 $\rightarrow a(1 A \_)(\_ z 2)$   
 $\rightarrow a(1 A \_)(\_ z 2)$   
(0z3)  $\rightarrow a(1 A \_)(\_ z 3)$   
 $\rightarrow a(1 A \_)(\_ z 3)$   
 $\rightarrow a(1 A \_)(\_ z 3)$   
 $\rightarrow a(1 A \_)(\_ z 3)$

### Step 3-5 ตรงกลางได้เลขตาม state ทั้งหมดเหมือนกัน

(0z0)  $\rightarrow$  a(1 A 0)(0 z 0)  
 $\rightarrow$  a(1 A 1)(1 z 0)  
 $\rightarrow$  a(1 A 2)(2 z 0)  
 $\rightarrow$  a(1 A 3)(3 z 0)

(0z1)  $\rightarrow$  a(1 A 0)(0 z 1)  
 $\rightarrow$  a(1 A 1)(1 z 1)  
 $\rightarrow$  a(1 A 2)(2 z 1)  
 $\rightarrow$  a(1 A 3)(3 z 1)

(0z2)  $\rightarrow$  a(1 A 0)(0 z 2)  
 $\rightarrow$  a(1 A 1)(1 z 2)  
 $\rightarrow$  a(1 A 2)(2 z 2)  
 $\rightarrow$  a(1 A 3)(3 z 2)

(0z3)  $\rightarrow$  a(1 A 0)(0 z 3)  
 $\rightarrow$  a(1 A 1)(1 z 3)  
 $\rightarrow$  a(1 A 2)(2 z 3)  
 $\rightarrow$  a(1 A 3)(3 z 3)

ทำกับ transition อื่นๆให้ครบ

**Transition**  $\delta(1, a, A) = (1, AA)$

(1A0)  $\rightarrow$  a(1 A 0)(0 A 0)  
 $\rightarrow$  a(1 A 1)(1 A 0)  
 $\rightarrow$  a(1 A 2)(2 A 0)  
 $\rightarrow$  a(1 A 3)(3 A 0)

(1A1)  $\rightarrow$  a(1 A 0)(0 A 1)  
 $\rightarrow$  a(1 A 1)(1 A 1)  
 $\rightarrow$  a(1 A 2)(2 A 1)  
 $\rightarrow$  a(1 A 3)(3 A 1)

(1A2)  $\rightarrow$  a(1 A 0)(0 A 2)  
 $\rightarrow$  a(1 A 1)(1 A 2)  
 $\rightarrow$  a(1 A 2)(2 A 2)  
 $\rightarrow$  a(1 A 3)(3 A 2)

(1A3)  $\rightarrow$  a(1 A 0)(0 A 3)  
 $\rightarrow$  a(1 A 1)(1 A 3)  
 $\rightarrow$  a(1 A 2)(2 A 3)  
 $\rightarrow$  a(1 A 3)(3 A 3)

### Result Step 1 - 3

$$S \rightarrow 0z0 \mid 0z1 \mid 0z2 \mid 0z3$$

$$(1A2) \rightarrow b$$

$$(2A2) \rightarrow b$$

$$(2z3) \rightarrow \lambda$$

$$(0z0) \rightarrow a(1A0)(0z0)$$

$$\rightarrow a(1A1)(1z0)$$

$$\rightarrow a(1A2)(2z0)$$

$$\rightarrow a(1A3)(3z0)$$

$$(0z1) \rightarrow a(1A0)(0z1)$$

$$\rightarrow a(1A1)(1z1)$$

$$\rightarrow a(1A2)(2z1)$$

$$\rightarrow a(1A3)(3z1)$$

$$(0z2) \rightarrow a(1A0)(0z2)$$

$$\rightarrow a(1A1)(1z2)$$

$$\rightarrow a(1A2)(2z2)$$

$$\rightarrow a(1A3)(3z2)$$

$$(0z3) \rightarrow a(1A0)(0z3)$$

$$\rightarrow a(1A1)(1z3)$$

$$\rightarrow a(1A2)(2z3)$$

$$\rightarrow a(1A3)(3z3)$$

$$(1A0) \rightarrow a(1A0)(0A0)$$

$$\rightarrow a(1A1)(1A0)$$

$$\rightarrow a(1A2)(2A0)$$

$$\rightarrow a(1A3)(3A0)$$

$$(1A1) \rightarrow a(1A0)(0A1)$$

$$\rightarrow a(1A1)(1A1)$$

$$\rightarrow a(1A2)(2A1)$$

$$\rightarrow a(1A3)(3A1)$$

$$(1A2) \rightarrow a(1A0)(0A2)$$

$$\rightarrow a(1A1)(1A2)$$

$$\rightarrow a(1A2)(2A2)$$

$$\rightarrow a(1A3)(3A2)$$

$$(1A3) \rightarrow a(1A0)(0A3)$$

$$\rightarrow a(1A1)(1A3)$$

$$\rightarrow a(1A2)(2A3)$$

$$\rightarrow a(1A3)(3A3)$$



#### Step 4 เอา production จาก step 2 ไปค้นหาใน step 3

$S \rightarrow 0z0 \mid 0z1 \mid 0z2 \mid 0z3$

$(1A2) \rightarrow b$

$(2A2) \rightarrow b$

$(2z3) \rightarrow \lambda$

$(0z0) \rightarrow a(1A0)(0z0)$

$\rightarrow a(1A1)(1z0)$

$\rightarrow a(1A2)(2z0)$

$\rightarrow a(1A3)(3z0)$

$(0z1) \rightarrow a(1A0)(0z1)$

$\rightarrow a(1A1)(1z1)$

$\rightarrow a(1A2)(2z1)$

$\rightarrow a(1A3)(3z1)$

$(0z2) \rightarrow a(1A0)(0z2)$

$\rightarrow a(1A1)(1z2)$

$\rightarrow a(1A2)(2z2)$

$\rightarrow a(1A3)(3z2)$

$(0z3) \rightarrow a(1A0)(0z3)$

$\rightarrow a(1A1)(1z3)$

$\rightarrow a(1A2)(2z3)$

$\rightarrow a(1A3)(3z3)$

$(1A0) \rightarrow a(1A0)(0A0)$

$\rightarrow a(1A1)(1A0)$

$\rightarrow a(1A2)(2A0)$

$\rightarrow a(1A3)(3A0)$

$(1A1) \rightarrow a(1A0)(0A1)$

$\rightarrow a(1A1)(1A1)$

$\rightarrow a(1A2)(2A1)$

$\rightarrow a(1A3)(3A1)$

$(1A2) \rightarrow a(1A0)(0A2)$

$\rightarrow a(1A1)(1A2)$

$\rightarrow a(1A2)(2A2)$

$\rightarrow a(1A3)(3A2)$

$(1A3) \rightarrow a(1A0)(0A3)$

$\rightarrow a(1A1)(1A3)$

$\rightarrow a(1A2)(2A3)$

$\rightarrow a(1A3)(3A3)$

### Step 5 ท production ที่ถูกไฮไลท์ทั้งหมด

$S \rightarrow 0z0 \mid 0z1 \mid 0z2 \mid 0z3$

$(1A2) \rightarrow b$

$(2A2) \rightarrow b$

$(2z3) \rightarrow \lambda$

$(0z0) \rightarrow a(1A0)(0z0)$

$\rightarrow a(1A1)(1z0)$

$\rightarrow a(1A2)(2z0)$

$\rightarrow a(1A3)(3z0)$

$(0z1) \rightarrow a(1A0)(0z1)$

$\rightarrow a(1A1)(1z1)$

$\rightarrow a(1A2)(2z1)$

$\rightarrow a(1A3)(3z1)$

$(0z2) \rightarrow a(1A0)(0z2)$

$\rightarrow a(1A1)(1z2)$

$\rightarrow a(1A2)(2z2)$

$\rightarrow a(1A3)(3z2)$

$(0z3) \rightarrow a(1A0)(0z3)$

$\rightarrow a(1A1)(1z3)$

$\rightarrow a(1A2)(2z3)$

$\rightarrow a(1A3)(3z3)$

$(1A0) \rightarrow a(1A0)(0A0)$

$\rightarrow a(1A1)(1A0)$

$\rightarrow a(1A2)(2A0)$

$\rightarrow a(1A3)(3A0)$

$(1A1) \rightarrow a(1A0)(0A1)$

$\rightarrow a(1A1)(1A1)$

$\rightarrow a(1A2)(2A1)$

$\rightarrow a(1A3)(3A1)$

$(1A2) \rightarrow a(1A0)(0A2)$

$\rightarrow a(1A1)(1A2)$

$\rightarrow a(1A2)(2A2)$

$\rightarrow a(1A3)(3A2)$

$(1A3) \rightarrow a(1A0)(0A3)$

$\rightarrow a(1A1)(1A3)$

$\rightarrow a(1A2)(2A3)$

$\rightarrow a(1A3)(3A3)$

### Step 6 production ที่เหลือของ step 3 ไม่เอาแล้ว ลบทั้งหมด

**Result**  $S \rightarrow 0z0 \mid 0z1 \mid 0z2 \mid 0z3$

$(1A2) \rightarrow b$

$(2A2) \rightarrow b$

$(2z3) \rightarrow \lambda$

$(0z3) \rightarrow a(1A2)(2z3)$

$(1A2) \rightarrow a(1A2)(2A2)$

**Step 7** เอา production ที่เหลือของ step 3 ไปเชิดกับ step 1, ที่เหลือลบทิ้ง

$$S \rightarrow 0z0 \mid 0z1 \mid 0z2 \mid 0z3$$

$$(1A2) \rightarrow b$$

$$(2A2) \rightarrow b$$

$$(2z3) \rightarrow \lambda$$

$$0z3 \rightarrow a(1A2)(2z3)$$

$$(1A2) \rightarrow a(1A2)(2A2)$$

**Result**

$$S \rightarrow 0z3$$

$$(1A2) \rightarrow b$$

$$(2A2) \rightarrow b$$

$$(2z3) \rightarrow \lambda$$

$$(0z3) \rightarrow a(1A2)(2z3)$$

$$(1A2) \rightarrow a(1A2)(2A2)$$

**Step 8** เปลี่ยนชื่อให้มันดูดีหน่อย

$$A = (0z3) \quad B = (1A2) \quad C = (2A3) \quad D = (2A2)$$

**Result**

$$S \rightarrow A$$

$$A \rightarrow aBC$$

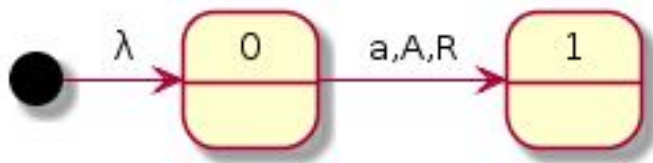
$$B \rightarrow aBD \mid b$$

$$D \rightarrow b$$

$$C \rightarrow \lambda$$

## Turing Machine

คือ automaton สำหรับเครื่องอ่านเทป



จากรูปหมายความว่า ถ้าอยู่ state 0 และอ่านเจอ **a** จะเขียน **A** ลงไปและเลื่อนไปทางขวา (R)

Transition = (input, output, direction)

## Type of Turing Machine

### 1. Acceptor

ใช้สำหรับตรวจสอบว่า input ในเทป ตรงกับ grammar หรือไม่

จำเป็นต้องมี final state

\*\* final state ของ turing machine เมื่อเข้าแล้วเครื่องจะหยุดทำงานทันที

### 2. Transducer

ใช้สำหรับอ่าน input และเขียน output ออกไป

โดยปกติแล้ว ไม่ต้องมี final state ก็ได้

ความรู้หมดแล้ววว