Information Science and Technology College of
Northeast Normal University

Franklin's 13 virtues and Puritan ethics

Temperance Silence Order Resolution Frugality Industry Sincerity Justice

(9)Moderation :Avoid extremes; forbear resenting injuries so much as you think they deserve.

中庸：为人处事不要极端偏激，对待他人的一时发泄，应加以宽容，不要斤斤计较、怀恨记仇。

"邓宁-克鲁格效应"：如果你很蠢，你就发现不了自己的蠢，因为发现自己的蠢需要相当高的智力。

(10) Cleanliness: Tolerate no uncleanliness in body, cloths, or habitation.

清洁：注重仪表，身体、衣服和住所都应当力求清洁。

Compiling and Running of Program

Dr. Zheng Xiaojuan
Professor

November. 2019

§ 5 Bottom-up Parsing

Information Science and Technology College of Northeast Normal University

The process to construct LR(0) Automata

V_T = {a, b, c}

V_N = {S, A, B}

S = S

P:

{ S' → S

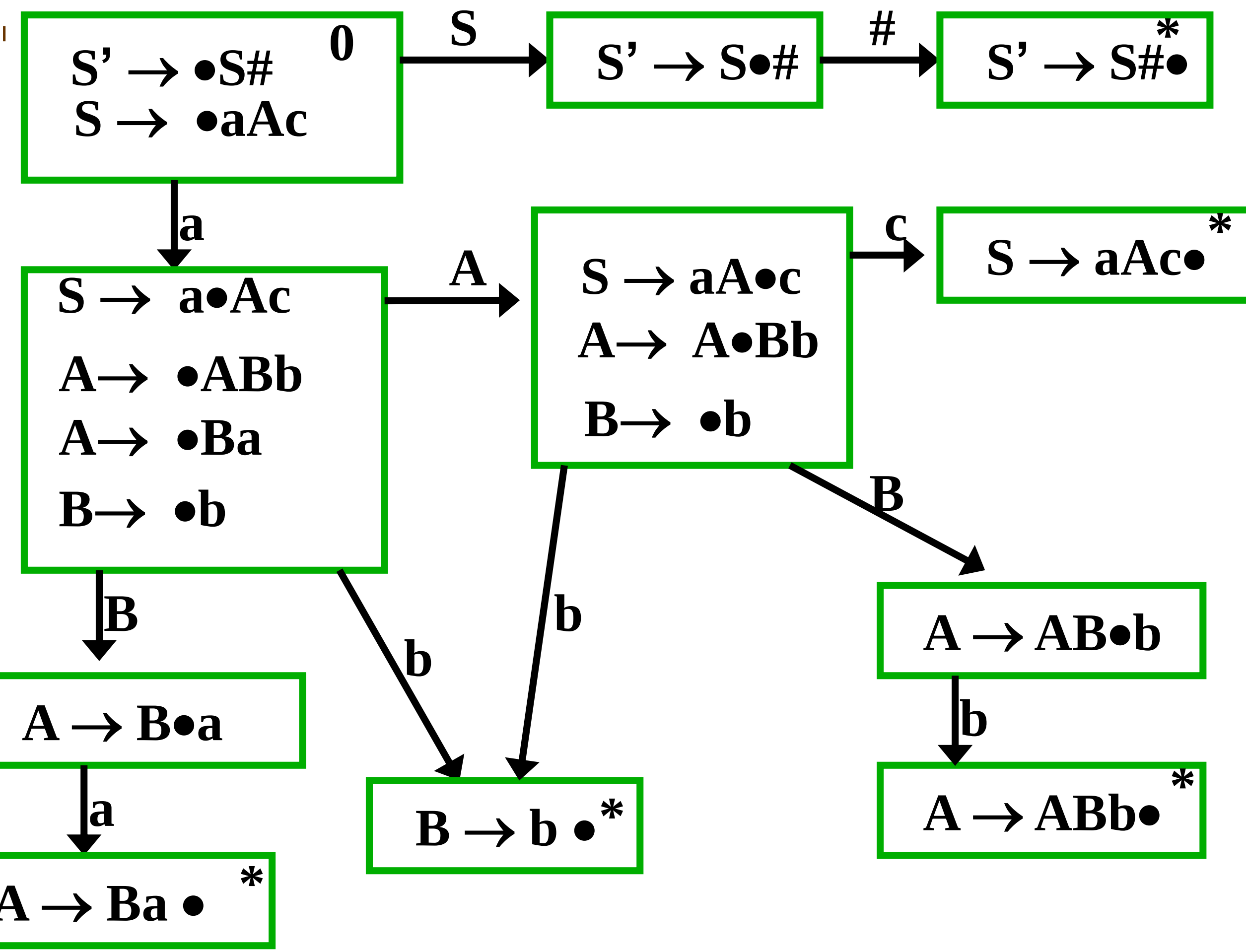
S → aAc

A → ABb

A → Ba

B → b

}



可归约活前缀集合为{S, aAc, aABb, aBa, aAb,ab}

- **Conclusion:**
  - **一个CFG的LR(0)自动机接受的是该CFG的所有*<u>可归约规范活前缀</u>*;**

5.3 LR(0) Parsing

- **LR(0) automata related concepts**
- **LR(0) grammar**
- **LR(0) Parsing Method**
  - **LR(0) Parsing Table**
  - **LR(0) Parsing Engine(驱动程序)**
- **LR(0) Parsing Process**

LR(0) automata related concepts

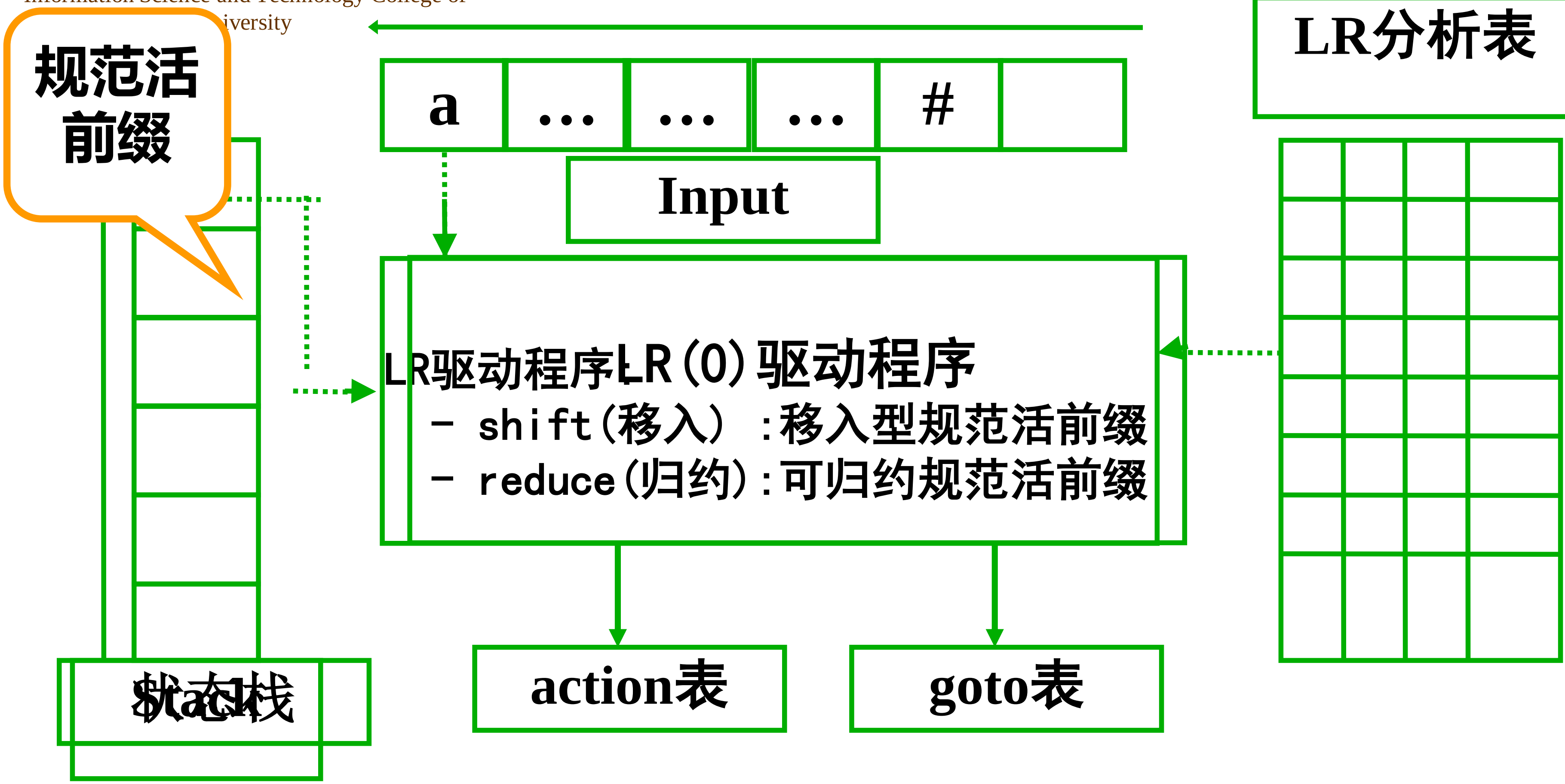
- 移入项目：$A \rightarrow \alpha \bullet a\beta$， $a \in V_T$
- 归约项目：$A \rightarrow \alpha \bullet$,
- 接受项目：$S' \rightarrow S \bullet \#$，（$S' \rightarrow S\#$是增广产生式）
- 待约项目：$A \rightarrow \alpha \bullet B\beta$，$B \in V_N$
- 移入状态:包含移入项目的状态(项目集)
- 归约状态:包含归约项目的状态(项目集)
- 冲突状态:
  - reduce – reduce conflict , 不同的归约项目
  - Shift – reduce conflict, 同时移入项目和归约项目

LR(0) grammar

- **Given a CFG *G***
- **LR_0 is LR(0) automata for *G***
- **If there is <u>no conflict</u> in any states of LR_0, *G* is called <u>*LR(0) grammar*</u>;**
  - 任意状态或者是移入状态，或者是归约状态
  - 如果是归约状态，一定存在一个唯一的归约项目，该归约项目对应一个产生式p，因此，该归约状态称为*p-归约状态*

Information Science and Technology College of ...iversity

LR(0) Parsing Method



LR(0) Grammar

$V_T = \{a, b, c\}$

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

$S = S$

P:

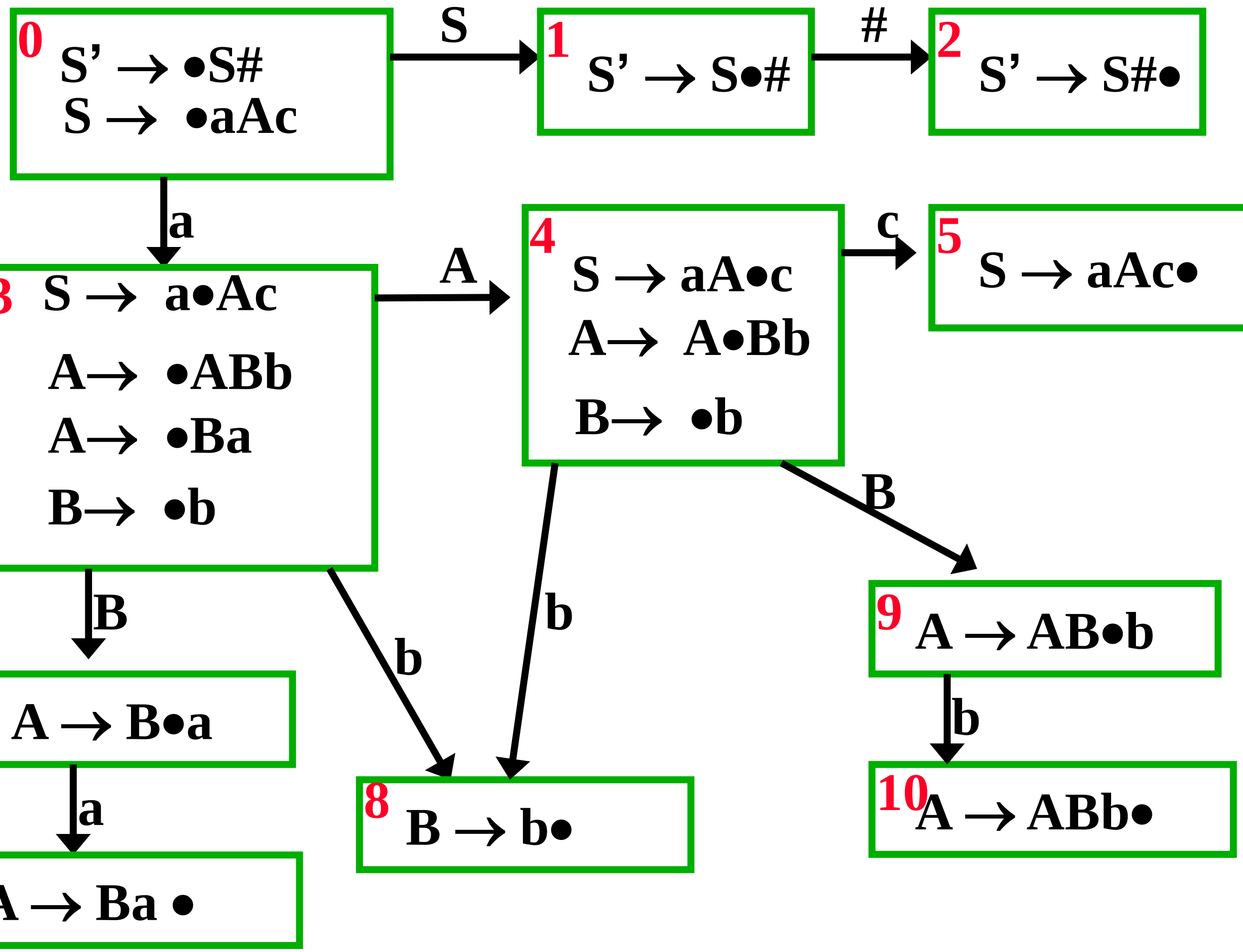
{ (1) $S \rightarrow aAc$

(2) $A \rightarrow ABb$

(3) $A \rightarrow Ba$

(4) $B \rightarrow b$

}



非LR(0) Grammar

$V_T = \{a, b, c\}$

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

$S = S$

P:

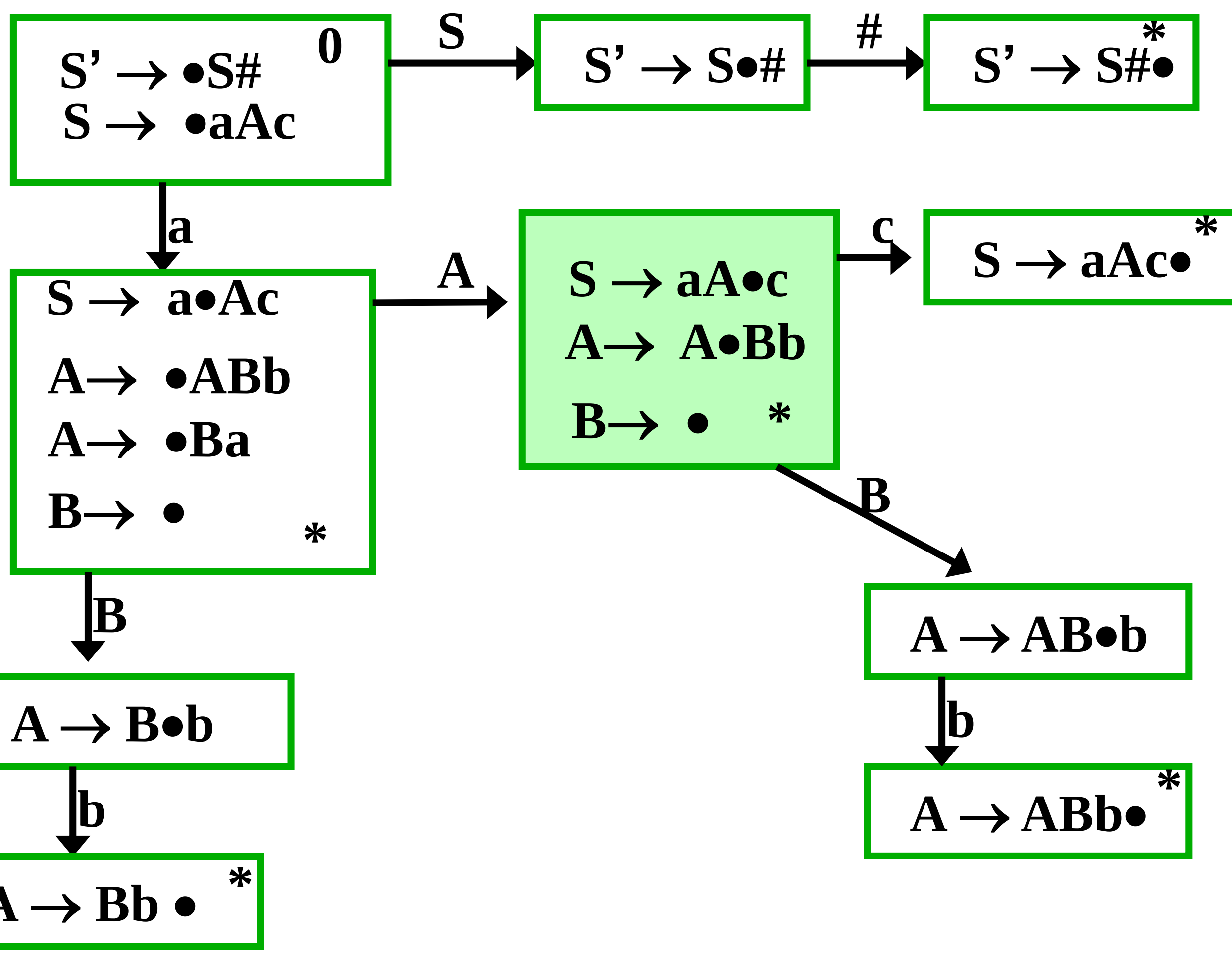
$\{ \; S \rightarrow aAc$

$A \rightarrow ABb$

$A \rightarrow Ba$

$B \rightarrow \varepsilon$

$\}$



LR(0) Parsing Table

- **action表**

- **goto表**

Information Science and Technology College of
Northeast Normal University

LR(0) Parsing Table

- **action表**

终极符 状　态	a_1	…	#
S_1			
…			
S_n			

action(S_i,a) = S_j, 如果S_i到S_j有a输出边

action(S_i,a) = R_p, 如果S_i是p−归约状态

action(S_i,#) = accept, 如果S_i是接受状态

action(S_i,a) = error, 其他情形

LR(0) Parsing Table

- **goto表**

非终极符 / 状 态	A_1	…	
S_1			
…			
S_n			

goto $(S_i, A) = S_j$, 如果S_i到S_j有A输出边

goto (S_i, A) = error,如果S_i没有A输出边

Information Science and Technology College of
Northeast Normal University

Example(Parsing Table)

V_T = {a, b, c}

V_N = {S, A, B}

S = S

P:

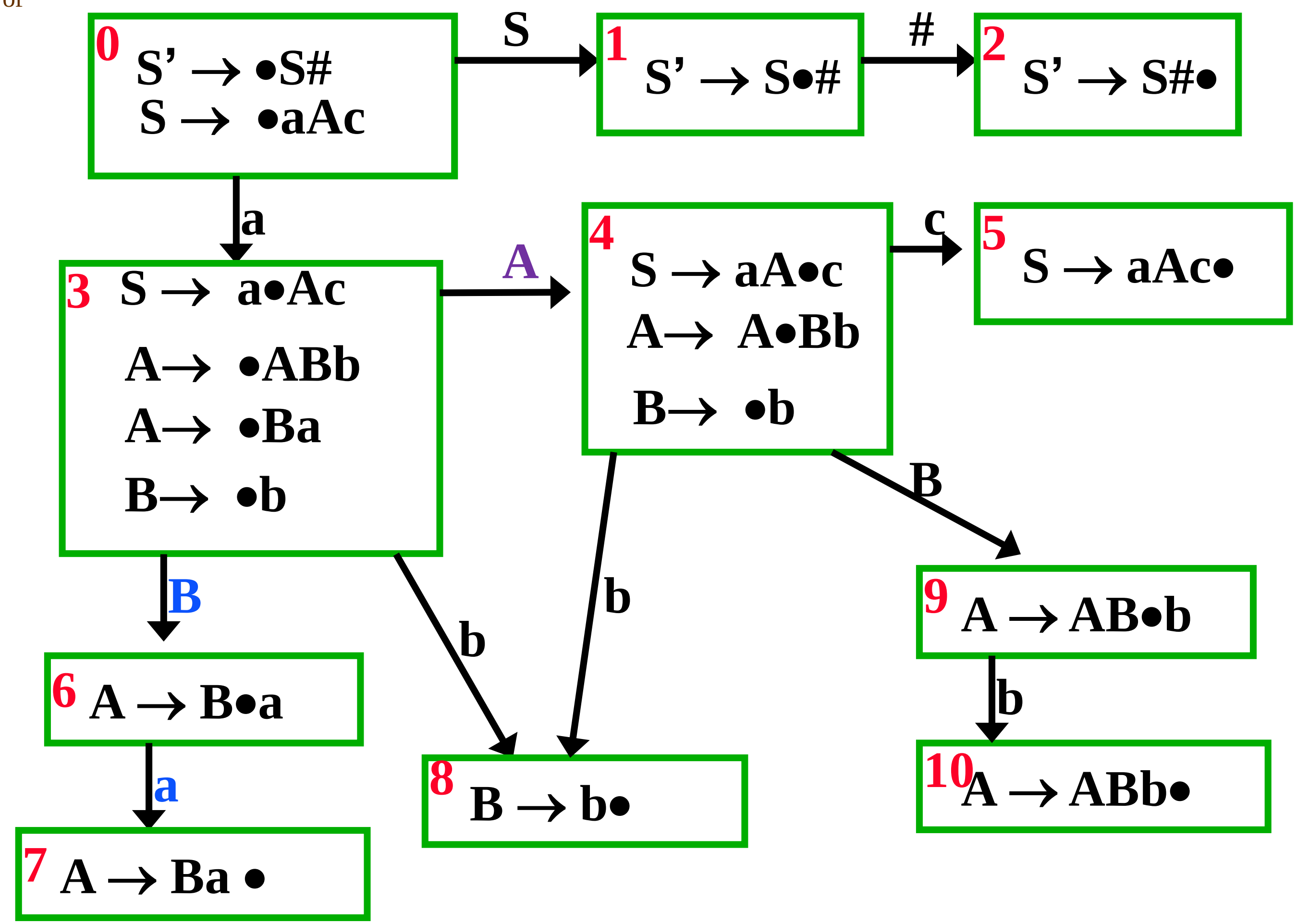
{ (1) S→ aAc

(2)A → ABb

(3)A → Ba

(4)B → b

}



Infor
North

分析表

	a	b	c	#
0	S3			
1				accept
2				
3		S8		
4		S8	S5	
5	R1	R1	R1	R1
6	S7			
7	R3	R3	R3	R3
8	R4	R4	R4	R4
9		S10		
10	R2	R2	R2	R2

	S	A	B
0	1		
1			
2			
3		4	6
4			9
5			
6			
7			
8			
9			
10			

(0) $S' \to S$
(1) $S \to aAc$
(2) $A \to ABb$
(3) $A \to Ba$
(4) $B \to b$

LR(0) Parsing Engine

- **Notations**
  - **S0: start state**
  - **Stack:状态栈**
  - **Stack(top):栈顶元素**
  - **P:产生式**
  - **| P |:产生式P右部符号个数;**
  - **P_A:产生式P左部非终极符;**
  - **Push(S):把状态S压入stack;**
  - **Pop(n):从stack弹出n个栈顶元素;**

LR(0) Parsing Engine

- **初始化: push(S0); a = readOne();**
- **L: Switch action(stack(top), a)**
  - **Case error: error();**
  - **Case accept: return true;**
  - **Case Si: push(Si), a=readOne(); goto L;**
  - **Case R_P: pop(|P|);**

    push(goto(stack(top), P_A));

    goto L;

LR(0) Parsing Process

P:

(0) S' → S
(1) S → aAc
(2) A → ABb
(3) A → Ba
(4) B → b

a	b	a	c

状态栈	输入流	分析动作
0	abac#	S3
03	bac#	S8
038	ac#	R4,Goto(3, B)=6
036	ac#	S7
0367	c#	R3, Goto(3, A)=4
034	c#	S5
0345	#	R1, Goto(0, S)=1
01	#	Accept

Assignment

$V_T = \{a, b, c, d\}$

$V_N = \{Z, A, B\}$

$S = Z$

P:

$\{Z \rightarrow ABd$

$A \rightarrow a$

$B \rightarrow d$

$B \rightarrow c$

$B \rightarrow bB$

$\}$

(1) 构造LR(0)分析表

(2) 给出abcd#的分析过程;

Information Science and Technology College of
Northeast Normal University

§ 5 Bottom-up Parsing

5.1 Overview of Bottom-up Parsing

5.2 Finite Automata for LR(0) Parsing

5.3 LR(0) Parsing

5.4 LR(1) Parsing

5.5 LALR(1) Parser Generator (YACC)

Let's See Why?

$V_T = \{a, b, =\}$

$V_N = \{S, L, R\}$

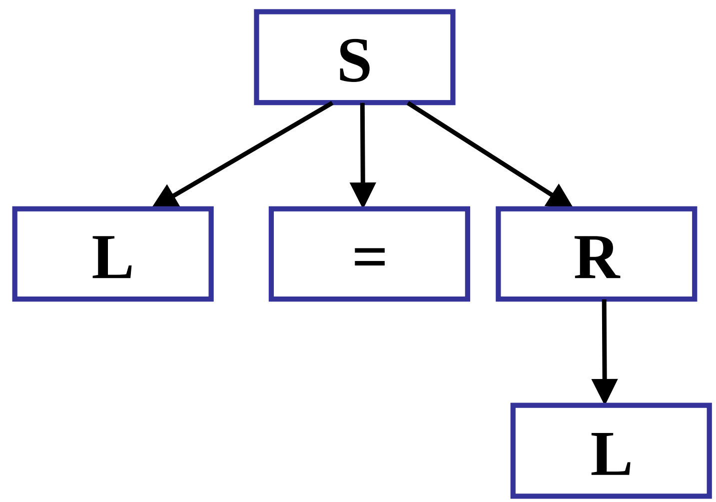
$S = S$

P:

{(1) $S \rightarrow L = R$
(2) $S \rightarrow R$
(3) $L \rightarrow aR$
(4) $L \rightarrow b$
(5) $R \rightarrow L$
}

- For the same non-terminal symbol, if it is in different position, its follow set might be different;

3	
$S \rightarrow L \bullet = R$	
$R \rightarrow L \bullet$	



One Solution

- **LR(1) Parsing**
  - **Basic idea:**
    - **对于非终极符的每个不同出现求其*后继终极符*，称为*<u>展望符</u>*;**
    - **对于一个非终极符的一个出现的所有后继终极符构成的集合称为*<u>展望符集</u>*;**
  - **Steps**
    - **Construct LR(1) automata**
      - **LR(1) item**
    - **Generate LR(1) parsing table (action & goto)**

5.4 LR(1) Parsing

- **Limitations of LR(0) Parsing**
- **LR(1) Automata**
- **LR(1) Parsing Table**
- **LR(1) Grammar**
- **LR(1) Parsing Process**

LR(1) Automata

- **LR(1) item**
  - **Two parts： (A→ α•β, {a, …})**
    - **(1) LR(0) item: A→ α•β**
    - **(2) 展望符集: {a, …}**
  - **Example**
    - **S→ L•=R , {#}**
    - **A → α• , {a, b}**
  - **展望符集的作用：**
    - **对于移入型项目，不起作用，但是需要保存；**
    - **对于归约型项目，表示只有当下一个输入符是其中一个展望符时，才可以进行归约动作；**

LR(1) Automata

- **LR(1) 项目集合*关于符号X的投影***
  - **IS is a set of LR(1) items;**
  - **X is a symbol;**
  - **$IS_{(X)}$ represents the projection of IS with respect to X:**
  - **$IS_{(X)} = \{(S \rightarrow \alpha X \bullet \beta, ss) \mid (S \rightarrow \alpha \bullet X \beta, ss) \in IS, X \in V_T \cup V_N\}$**
- **$IS = \{(A \rightarrow A \bullet Bb, \{a,b\}), (B \rightarrow a \bullet, \#), (B \rightarrow b \bullet B, \{b\})\}$**
- **$X = B$**
- **$IS_{(B)} = \{(A \rightarrow AB \bullet b, \{a, b\}), (B \rightarrow bB \bullet, \{b\})\}$**

LR(1) Automata

- LR(1)项目集合的*闭包*
  - IS is a set of LR(1) items;
  - CLOSURE(IS)是一个LR(1)项目集合，按照下面的步骤计算:

[1] 初始, CLOSURE(IS) = IS;

[2] 对于CLOSURE(IS)没有处理的LR(1)项目,

如果其形式为 $(B\rightarrow \beta\bullet A\pi,\ ss)$,

而且A的全部产生式是 $\{A\rightarrow\alpha1, \ldots, A\rightarrow\alpha n\}$

则增加如下LR(1)项目到CLOSURE(IS)

$\{(A\rightarrow \bullet\alpha1,\ ss'), \ldots, (A\rightarrow \bullet\alpha n,\ ss')\}$,

其中 ss' = first(π), 如果符号串π不导出空;

ss' = (first(π)-{ε}) ∪ss, 如果符号串π导出空;

[3] 重复[2]直到 CLOSURE(IS)收敛;

LR(1)自动机

- **goto函数**
  - **IS is a set of LR(1) items;**
  - **X is a symbol;**
  - **goto(IS, X) = CLOSURE(IS $_{(X)}$)**

$V_T = \{a, b, =\}$

$V_N = \{S, L, R\}$

S = S

P:

{(1) $S \rightarrow L = R$

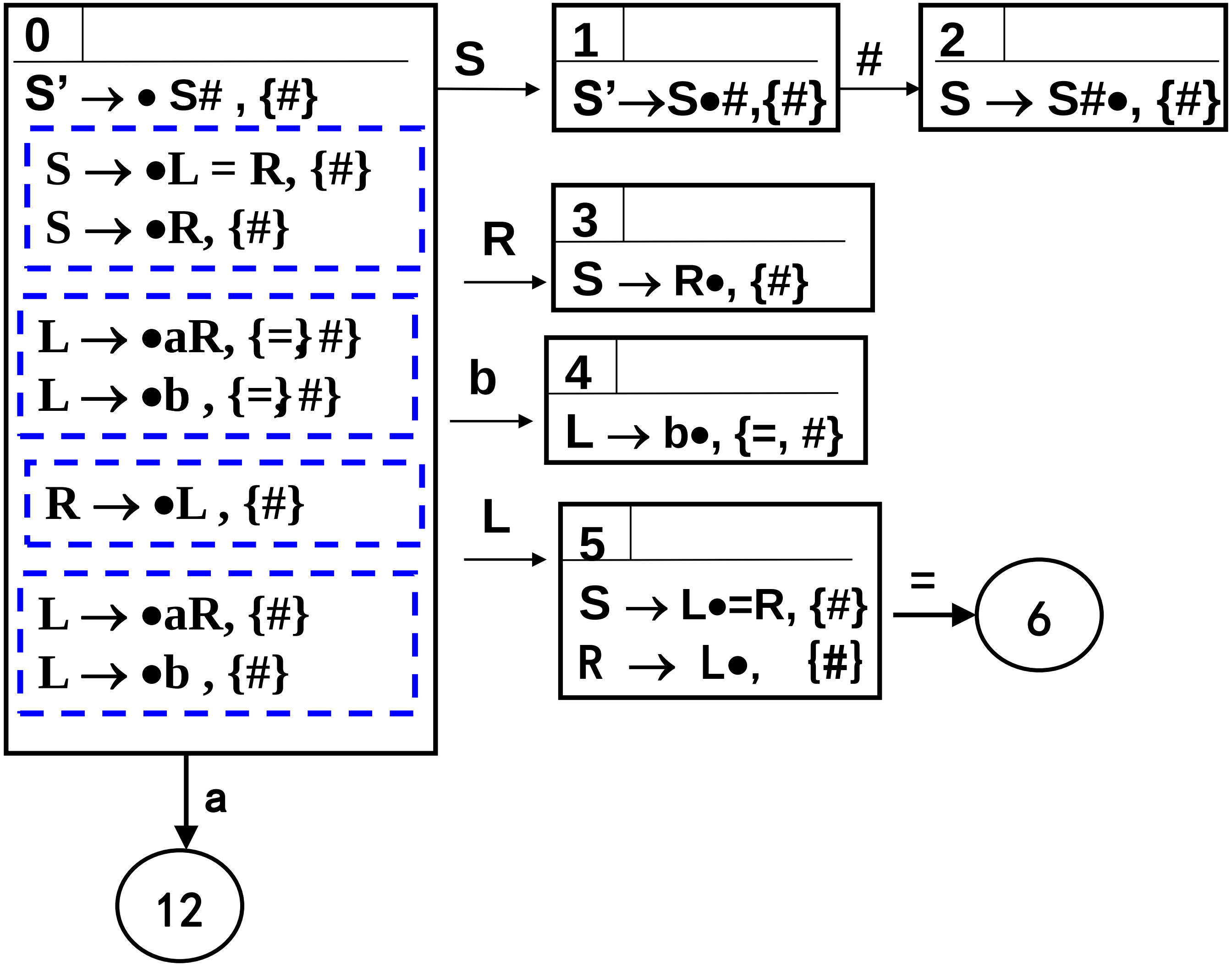
(2) $S \rightarrow R$

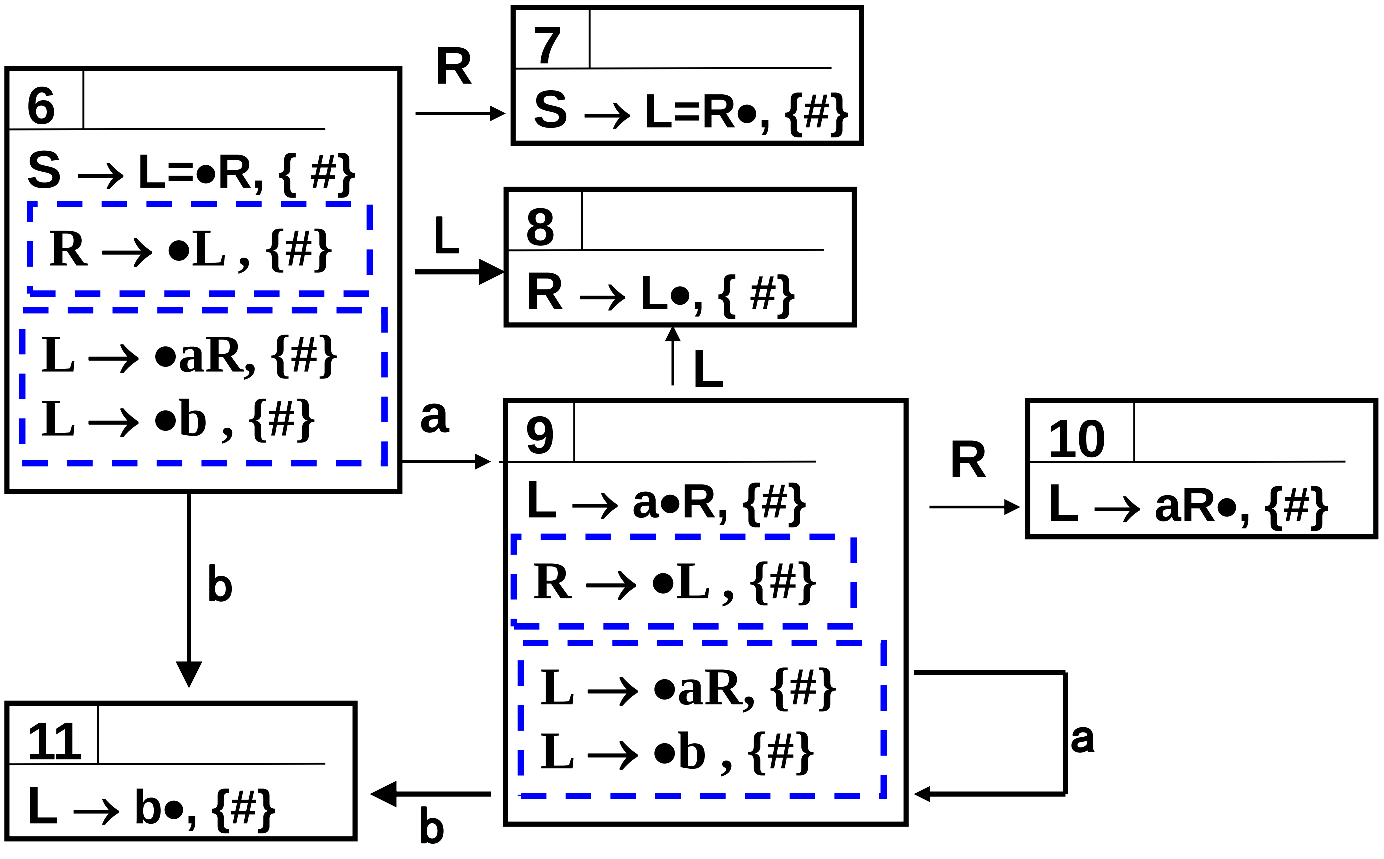
(3) $L \rightarrow aR$

(4) $L \rightarrow b$

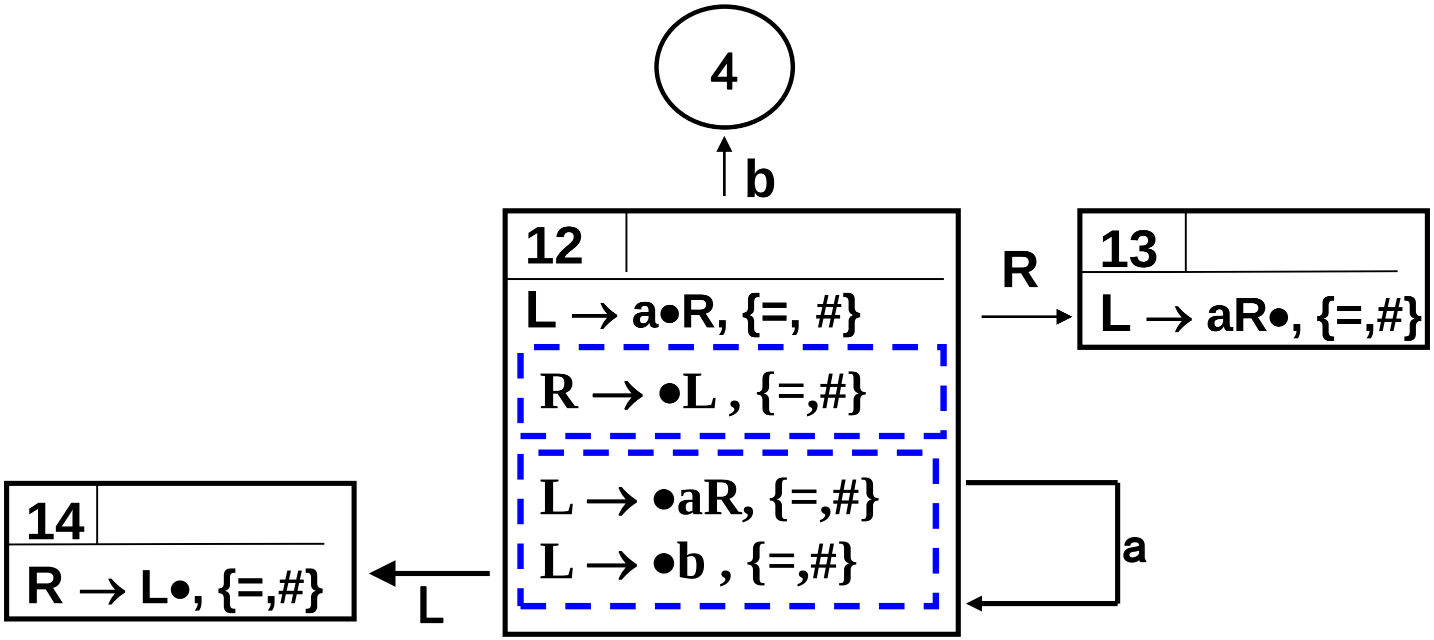
(5) $R \rightarrow L$

}





6
S → L=•R, { #}
R → •L , {#}
L → •aR, {#}
L → •b , {#}
R
7
S → L=R•, {#}
L
8
R → L•, { #}
L
a
9
L → a•R, {#}
R → •L , {#}
L → •aR, {#}
L → •b , {#}
R
10
L → aR•, {#}
a
b
11
L → b•, {#}
b



4
b
12
L → a•R, {=, #}
R → •L , {=,#}
L → •aR, {=,#}
L → •b , {=,#}
R
13
L → aR•, {=,#}
a
14
R → L•, {=,#}
L

如何计算展望符集?

- **投影得到的项目**
  - **继承**

$S\rightarrow \alpha\bullet X\beta, ss \xrightarrow{X} S\rightarrow \alpha X\bullet\beta, ss$

- **闭包新产生的项目**
  - **扩展**

$S\rightarrow \alpha\bullet A\beta, ss$

$A \rightarrow \bullet\alpha 1, ss'$

....

$A \rightarrow \bullet\ \alpha n, ss'$

....

ss' = first(β), 如果β不导出空;

ss' = (first(β)-{ε}) ∪ss, 如果β导出空;

Process of Constructing LR(1) Automata

- [1]增广产生式 $S' \rightarrow S\#$
- [2]$\Sigma = V_T \cup V_N \cup \{\#\}$
- [3]$S0 = CLOSURE(S' \rightarrow \bullet S)$
- [4]$ISS = \{S0\}$
- [5]对于ISS中的每一个项目集合IS，和每个符号$X \in \Sigma$，
  计算IS' = goto(IS，X)，
  如果IS'不为空，则 建立 $IS \xrightarrow{X} IS'$，
  如果IS'不为空且IS'不属于ISS,则把IS'加入ISS;
- [6]重复[5]直到ISS收敛;

LR(1) Parsing Table

- **action表**

- **goto表**

LR(1) Parsing Table

- **action表**

终极符 / 状 态	a_1	…	#
S_1			
…			
S_n			

(1) action(S_i, a) = S_j，如果S_i到S_j有a输出边

(2) action(S_i, a) = R_p，如果S_i中包含这样LR(1)项目，

(A→α•，ss)，其中A→α是产生式P，且a∈ss；

(3) action(S_i, #) = accept，如果S_i是接受状态

(4) action(S_i, a) = error，其他情形

LR(1) Parsing Table

- **goto表**

非终极符 / 状态	A_1	…	A_n
S_1			
…			
S_n			

goto（S_i，A）= S_j，如果S_i到S_j有A输出边

goto（S_i，A）= error，如果S_i没有A输出边

(1) S → L = R
(2) S → R
(3) L → aR
(4) L → b
(5) R → L

LR(1) Parsing Table

	Action table					Goto table		
	a	b	=	#		S	L	R
0	S12	S4				1	5	3
1				Accept				
2								
3				R2				
4			R4	R4				
5			S6	R5				
6	S9	S11					8	7
7				R1				

(1) S → L = R
(2) S → R
(3) L → aR
(4) L → b
(5) R → L

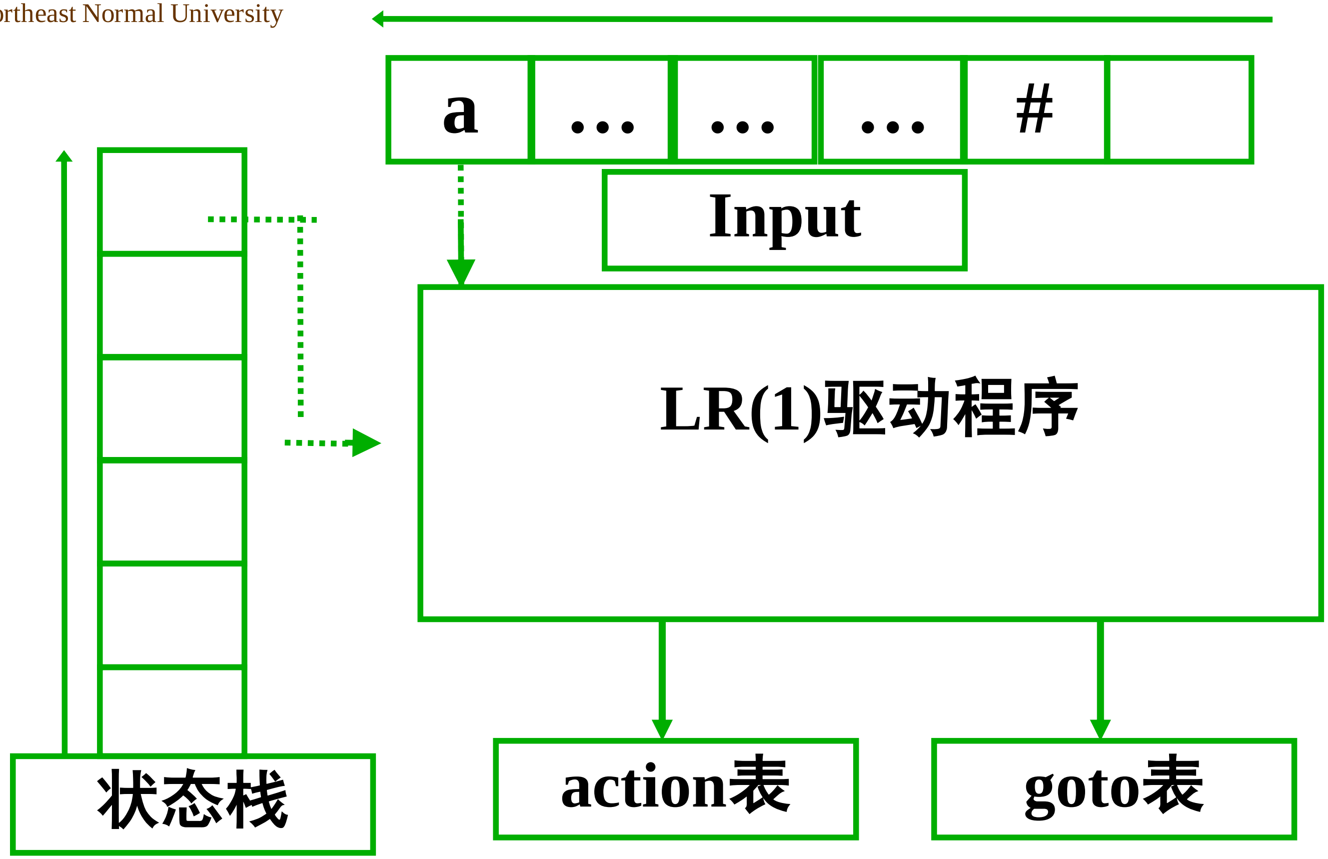
LR(1) Parsing Table (cond.)

	Action table					Goto table		
	a	b	=	#		S	L	R
8				R5				
9	S9	S11						10
10				R3				
11				R4				
12	S12	S4					14	13
13			R3	R3				
14			R4	R4				
15								

LR(1) Grammar

- **Given a CFG *G***
- **LR_1 is LR(1) automata for *G***
- ***A_1 is action table for G***
- **If there is only one action for one state and one terminal symbol in A_1, then G is called *LR(1) Grammar*;**
  - **Shift**
  - **Reduce**
  - **Accept**
  - **Error**

LR(1) Parsing Method



LR(1) Parsing Engine

- **初始化: push(S0); a = readOne();**
- **L: Switch action(stack(top), a)**
  - **Case *<u>error</u>*: error();**
  - **Case *<u>accept</u>*: return true;**
  - **Case *<u>Si</u>*: push(Si), a=readOne(); goto L;**
  - **Case $\underline{R_P}$: pop(|P|);**

    push(goto(stack(top), P_A));

    goto L;

How to generate parse tree during LR parsing?

The Process of LR(1) Parsing

V_T = {a, b, =}

V_N = {S, L, R}

S = S

P:
{(1) S → L = R
(2) S → R
(3) L → aR
(4) L → b
(5) R → L
}

状态栈	输入流	分析动作
0	b=b#	S4
04	=b#	R4, Goto(0,L)=5
05	=b#	S6
056	b#	S11
056(11)	#	R4, Goto(6, L)=8
0568	#	R5 , Goto(6, R)=7
0567	#	R1, , Goto(0, S)=1
01	#	Accept

5.5 LALR(1) Parser Generator (YACC)

- **Widely used for automatically generating parser for a programming language;**
- **It is difficult to solve conflicts if the grammar is not LALR(1) grammar;**
- **Please find out the general process of using YACC to generate a parser?**

5.5 LALR(1) Parser Generator (YACC)

- YACC——Yet Another Compiler Compiler
  - Stephen C. Johnson. YACC: Yet Another Compiler-Compiler. *Unix Programmer's Manual* Vol 2b, 1979.
  - LALR(1)分析
  - GNU Bison：基本兼容Yacc，与flex一起使用
  - Berkeley Yacc
- The Lex & Yacc Page
  - http://dinosaur.compilertools.net/

Assignment(1)

$V_T = \{0, 1\}$

$V_N = \{S\}$

S = S

P:

{(1) $S \rightarrow 0S1$

(2) $S \rightarrow 1S0$

(3) $S \rightarrow 10$

}

Please check whether this grammar is LR(1) Grammar?
Please give the process of how you produce the result.

Assignment(2)

(1)构造LR(1)自动机;
(2)构造LR(1)分析表;

V_T = { {, }, ;, id, +, (,), =, [,] }

V_N = {B, SL, S, V, E, T}

S = B

P:

{(1)B → { SL }   (2) SL → SL; S   (3) SL → S

(4)S → B   (5) S → V = E   (6) V→id

(7)V → id[E]   (8) E → E+T   (9) E → T

(10) T → V   (11) T → (E)

}

Answer to Assignments

Assignment(1)

$V_T = \{0, 1\}$

$V_N = \{S\}$

S = S

P:

$\{(1)\ S \rightarrow 0S1$

$(2)\ S \rightarrow 1S0$

$(3)\ S \rightarrow 10$

$\}$

Please check whether this grammar is LR(1) Grammar?

Please give the process of how you produce the result.

$V_T = \{0, 1\}$

$V_N = \{S\}$

S = S

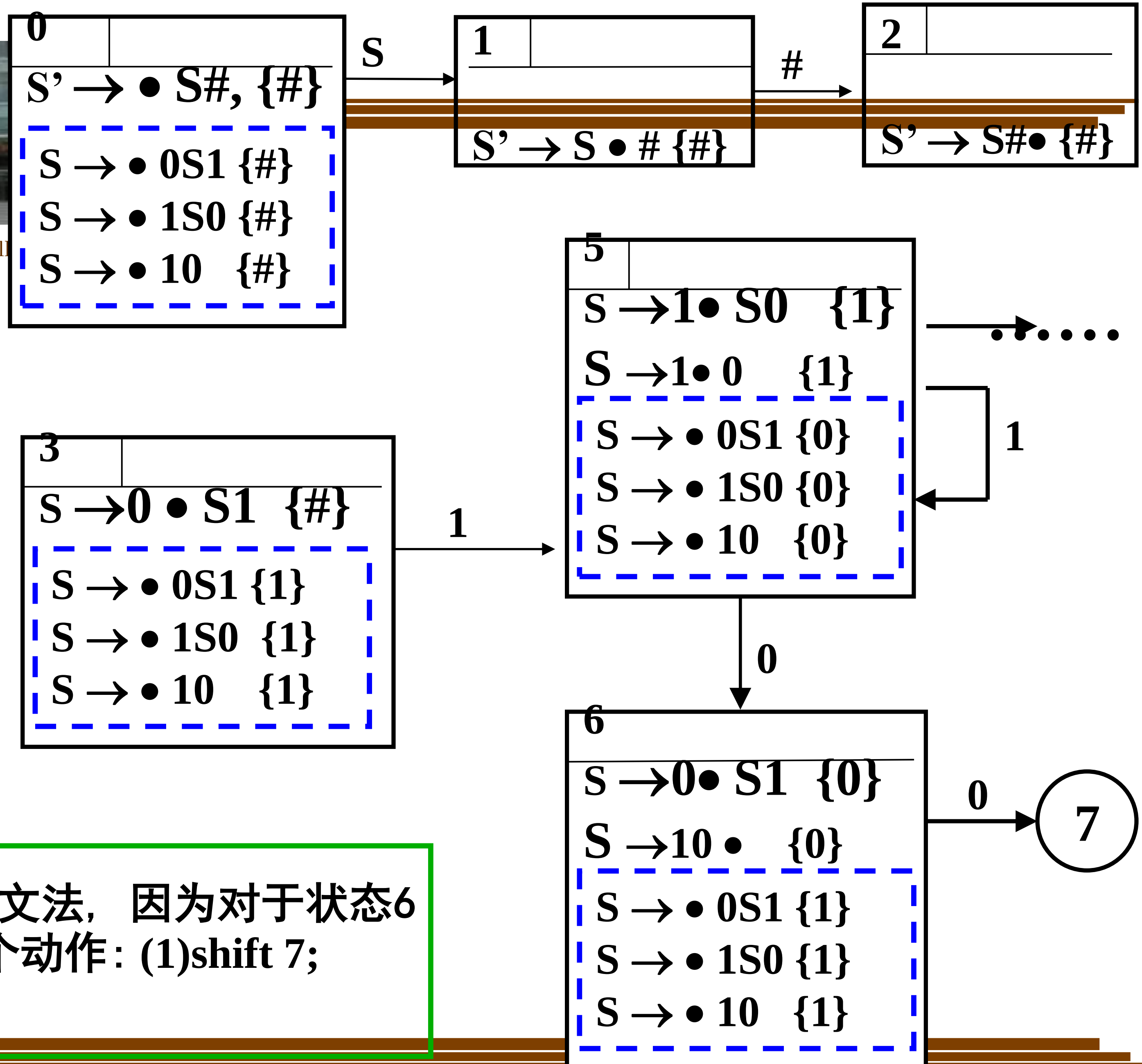
P:

{(1) S→ 0S1

(2) S→ 1S0

(3) S → 10

}



该文法不是LR(1)文法，因为对于状态6和终极符0，有两个动作：(1)shift 7; (2) Reduce 3;

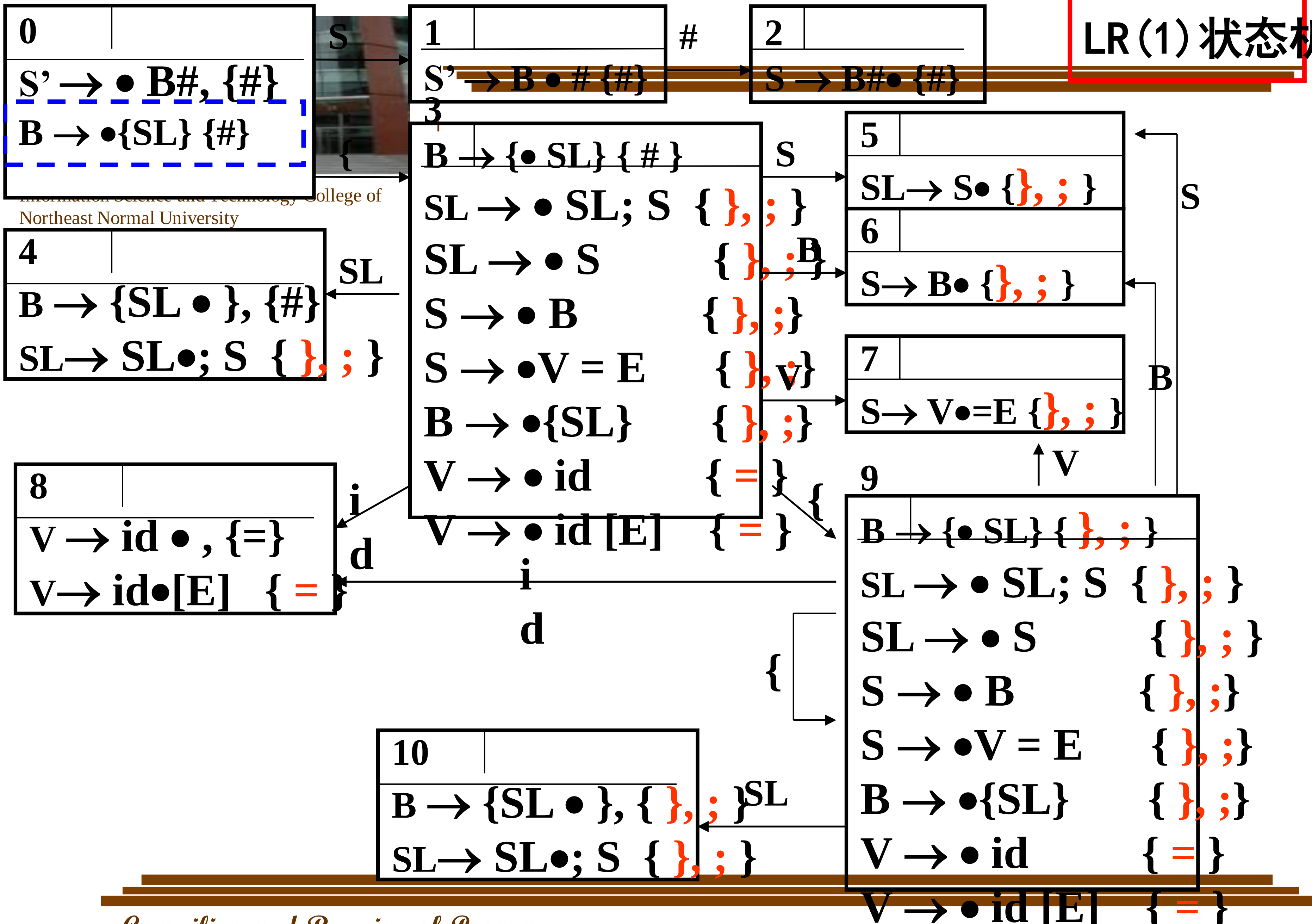
Assignment(2)

(1) 构造LR(1)自动机；
(2) 构造LR(1)分析表；

V_T = { {, }, ;, id, +, (,), =, [,] }
V_N = {B, SL, S, V, E, T}
S = B
P:

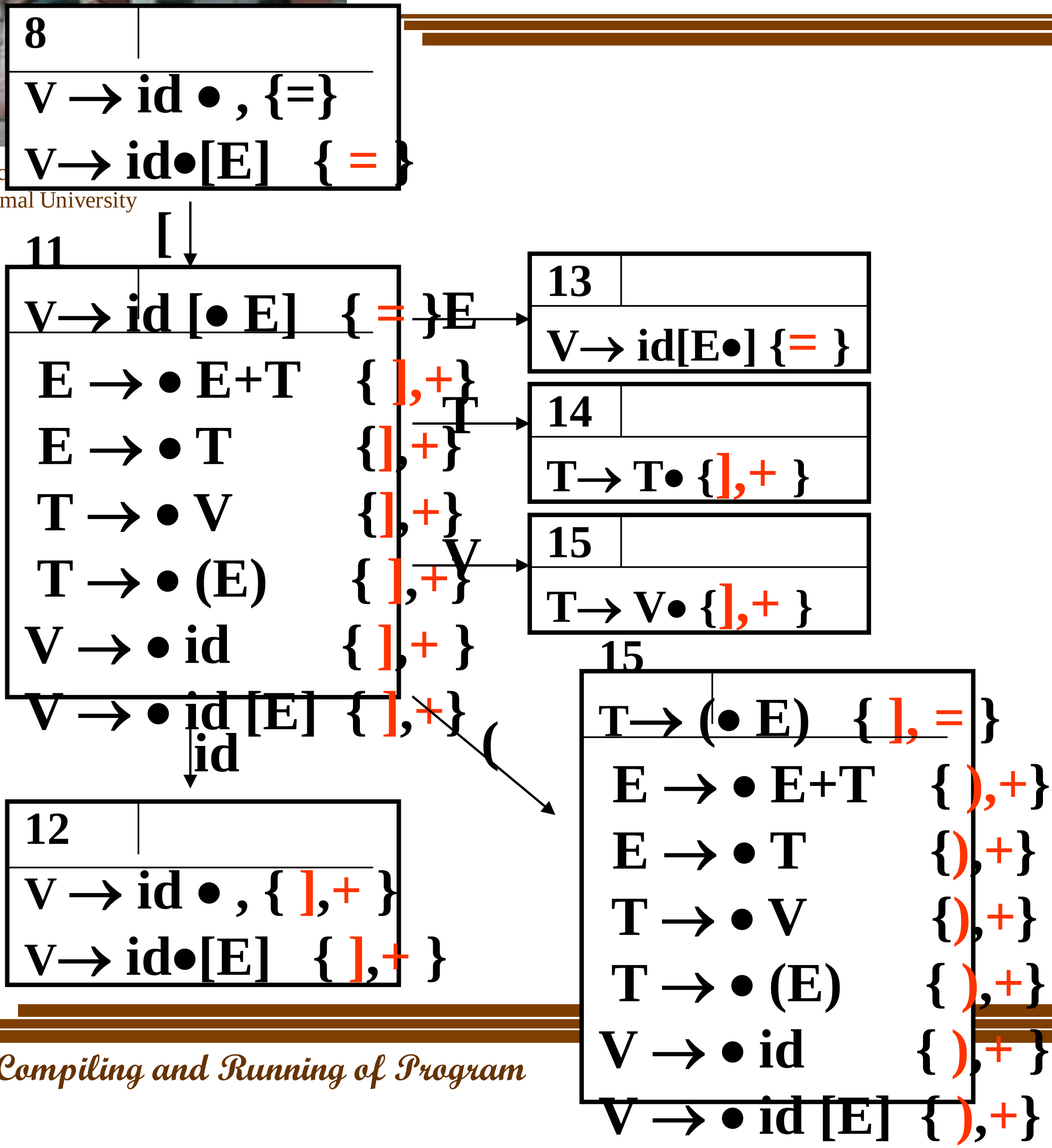
{(1) $B \rightarrow$ { SL }　(2) $SL \rightarrow SL; S$　(3) $SL \rightarrow S$
(4) $S \rightarrow B$　(5) $S \rightarrow V = E$　(6) $V \rightarrow id$
(7) $V \rightarrow id[E]$　(8) $E \rightarrow E+T$　(9) $E \rightarrow T$
(10) $T \rightarrow V$　(11) $T \rightarrow (E)$
}

LR(1)状态机



Information Science and Technology College of
Northeast Normal University

Information Sc
Northeast Normal University



7

S→ V•=E { }, ; }

=

16

S→ V = • E { }, ; }

E → • E+T { }, ;,+ }

E → • T { }, ;, +}

T → • V { }, ;, + }

T → • (E) { }, ;, + }

V → • id { }, ;,+ }

V → • id [E] { }, ;,+ }