Lecture 7: LR分析法

Xiaoyuan Xie 谢晓园
xxie@whu.edu.cn
计算机学院E301

7.1 LR(0)局限性

7.1 LR(0)分析的局限

- LR(0)文法仅凭符号栈里的内容来确定可归约活前缀, 非常容易产生冲突;
- LR(0)文法易于产生冲突的原因在于在确定分析动作时没有考虑输入串信息。
- 事实上,只有有限的文法能满足LR(0)文法的条件;
- LR(0)分析不是一种实用的方法，只是为介绍LR分析的思想而引进的；

LR(0)自动机的移入-归约冲突

$V_T = \{a, b\}$

$V_N = \{S, A\}$

S = S

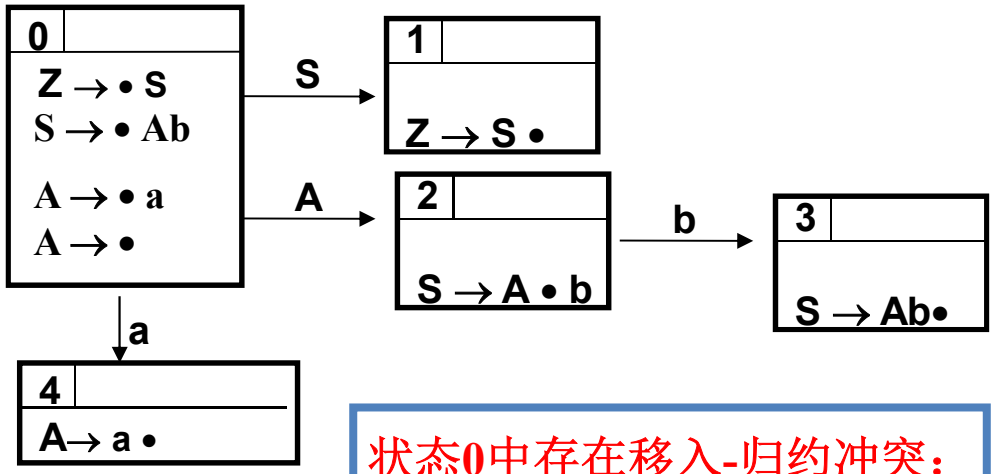
P:

{ (1) S→ Ab

(2) A → ε

(3) A → a

}



状态0中存在移入-归约冲突：

(1) 移入项目:A → • a

(2) 归约项目: A → •

LR(0)自动机的归约-归约冲突

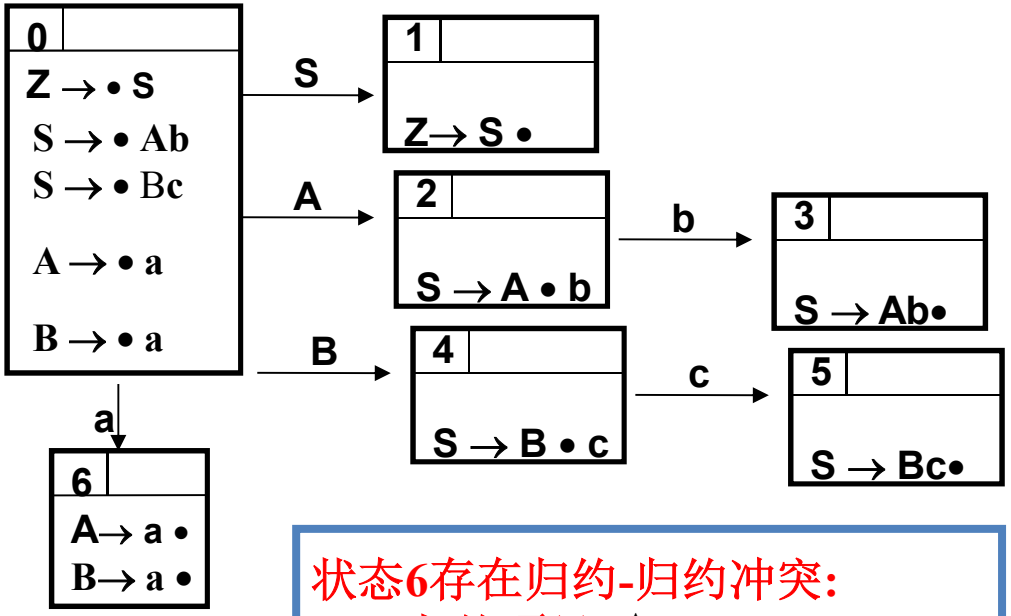
V_T = {a, b, c}

V_N = {S, A, B}

S = S

P:

{(1) S→ Ab
(2) S→ Bc
(3) A → a
(4) B → a
}



状态6存在归约-归约冲突:

(1) 归约项目: A → a •

(2) 归约项目: B → a•

如何消除冲突？

- **改进LR(0)**
  - SLR
  - LR(1)
  - LALR

7.2 SLR

改进策略之SLR

- **消除冲突：向前看一个输入符号来选择分析动作(考虑现实)**
  - 对于任何形如I={X→γ• bβ，A→ α• ，B→ α•}的LR(0)项目集，
    - 假设下一个输入符号是: a，则存在规约冲突；但
    - 如果Follow(A)∩Follow(B) = Φ, 且b∉Follow(A)，b∉Follow(B)，则可以通过如下方法进行冲突消除

改进策略之SLR

- **消除冲突：向前看一个输入符号来选择分析动作(考虑现实)**
  - 移入-归约冲突(S-R冲突)
    - 移入: 如存在移入项目 $A\rightarrow \alpha\bullet a\beta$
    - 归约: 如果存在归约项目 $B\rightarrow \pi\bullet$, 且 $a \in follow(B)$
  - 归约-归约冲突(R-R冲突)
    - 归约(P1): 如果存在归约项目 $A\rightarrow \pi\bullet$, $a\in follow(A)$, 产生式P1= $A\rightarrow \pi$
    - 归约(P2): 如果存在归约项目 $B\rightarrow \pi'\bullet$, $a\in follow(B)$, 产生式P2= $B\rightarrow\pi'$

LR(0)分析表 (带有S-R 冲突)

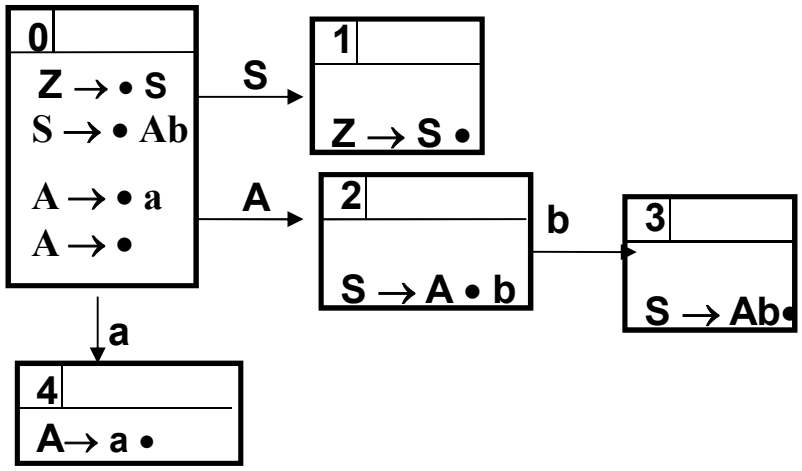
V_T = {a, b}

V_N = {S, A}

S = S

P:

{ (1) S→ Ab
(2) A → ε
(3) A → a
}



状态0中存在移入-归约冲突：
(1) 移入项目: A → • a
(2) 归约项目: A → •

	Action 表				Go to 表	
	a	b	#		S	A
0	S5;R2	R2	R2		1	3
1			Accept			
2		S3				
3	R1	R1	R1			
4	R3	R3	R3			

LR(0)分析表 (没有冲突)

$V_T = \{a, b\}$

$V_N = \{S, A\}$

S = S

P:

{ (1) S→ Ab

(2) A → ε

(3) A → a

}

	Action 表				Goto 表	
	a	b	#		S	A
0	S5	R2			1	3
1			Accept			
2		S3				
3			R1			
4		R3				

冲突用follow(A)解决掉了

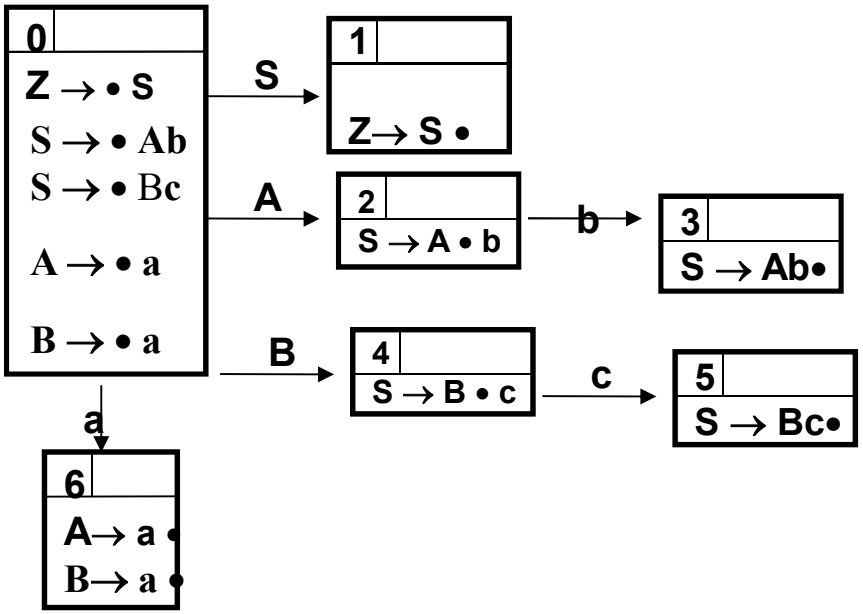
LR(0)分析表 (带有R-R 冲突)

V_T = {a, b, c}

V_N = {S, A, B}

S = S

P:
{(1) S→ Ab
(2) S→ Bc
(3) A → a
(4) B → a
}



	Action 表					Goto 表		
	a	b	c	#		S	A	B
0	S7					1	3	5
1				Accept				
2		S4						
3	R1	R1	R1	R1				
4			S6					
5	R2	R2	R2	R2				
6	R3 R4	R3 R4	R3 R4	R3 R4				

状态6存在归约-归约冲突:
(1) 归约项目: A → a •
(2) 归约项目: B → a•

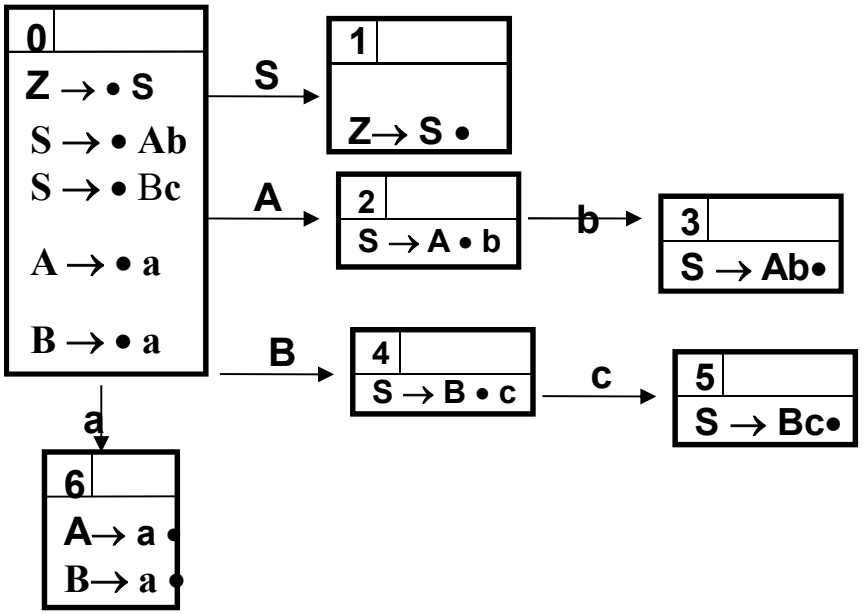
LR(0)分析表 (带有R-R 冲突)

V_T = {a, b, c}

V_N = {S, A, B}

S = S

P:
{(1) S→ Ab
(2) S→ Bc
(3) A → a
(4) B → a
}



	Action 表					Goto 表		
	a	b	c	#		S	A	B
0	S7					1	3	5
1				Accept				
2		S4						
3	R1	R1	R1	R1				
4			S6					
5	R2	R2	R2	R2				
6		R3	R4					

状态6存在归约-归约冲突:
(1) 归约项目: A → a •
(2) 归约项目: B → a•

冲突用 follow(A) 和 follow(B)消除了

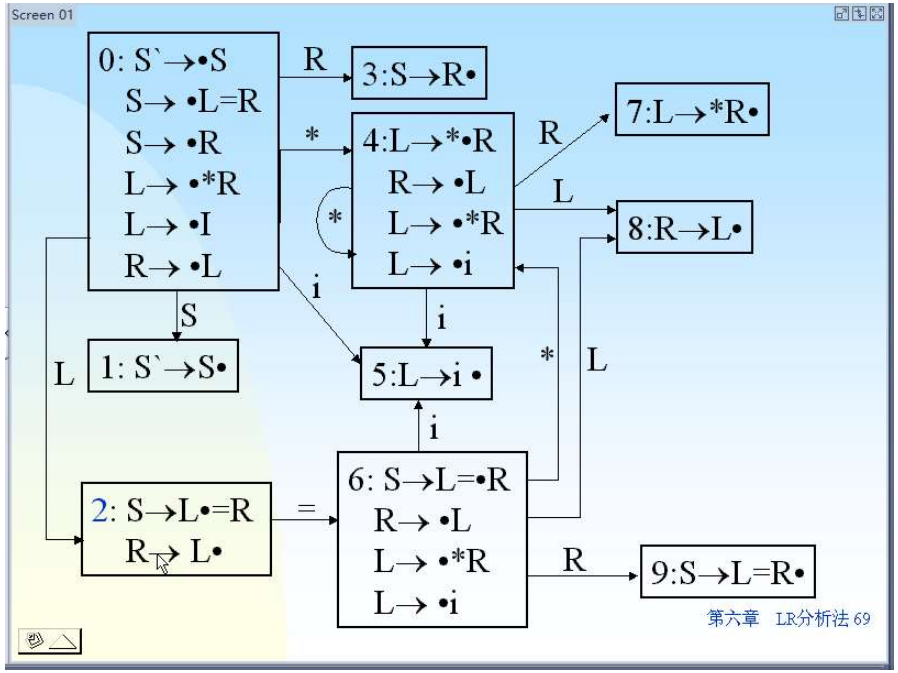
SLR(1) 分析

- **SLR(1) 分析的思想**
  - **向前看一个输入符号;**
  - **用非终极符的follow集合解决冲突;**
  - **如果能将LR(0)自动机中的所有冲突消除掉，则该文法称为*SLR(1)文法*，否则称为非SLR(1)文法。**
- **SLR(1)文法**
  - **文法G的*SLR(1)自动机*与它的LR(0)自动机完全相同**
  - ***SLR(1)分析表*中Action表与LR(0)分析表的Action表稍有不同(移入动作没有区别，归约动作有区别)**
  - ***SLR(1)驱动程序*与LR(0)驱动程序也相同**

非SLR(1)文法的例子

- 例：一个非SLR文法的例子。有如下文法:
- 1. S` →S　　2. S →L=R
- 3. S →R　　4. L →*R
- 5. L → i　　6. R →L

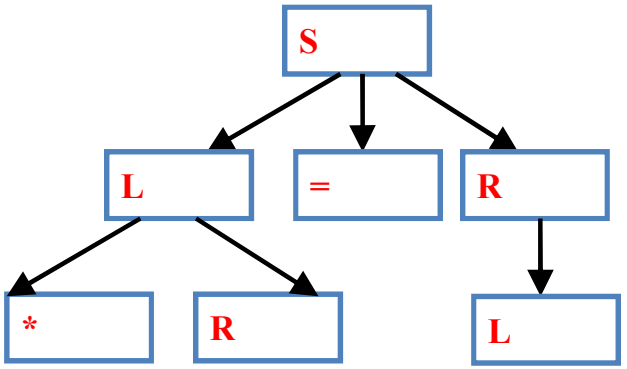
follow(R) = {#, =}



SLR(1)分析的局限性

- **SLR(1)分析存在的问题**
  - 通过引入follow集，解决了一部分文法的冲突问题，但满足SLR(1)文法条件的文法仍有限；
  - 原因在于：对于同一个非终极符可能出现在不同的位置，不同位置的后继符号(follow)是不同的，统一看待，仍有可能引起冲突。

- 例：一个非SLR文法的例子。有如下文法:
- 1. S` →S　　2. S →L=R
- 3. S →R　　4. L →*R
- 5. L → i　　6. R →L



SLR(1)分析的局限性

- 对SLR分析的思考
  - 在构造SLR分析表的方法中，若项目集I_k中含有$A\rightarrow\alpha\bullet$，那么在状态k时，只要面临 输入符号$a \in Follow(A)$，就确定采用$A\rightarrow\alpha$产生式进行归约。但是，在某种情况下，当状态k呈现于栈顶时，栈里的符号串所构成的活前缀$\beta\alpha$未必允许把α归约为A。因为可能没有一个规范句型含有前缀βAa。因此此时用$A\rightarrow\alpha$产生式进行归约未必有效。

7.3 LR(1)

解决的办法

- **结论**
  - 并非Follow集中的符号都会出现在规范句型中
- **对策**
  - 给每个LR(0)项目添加展望信息，即添加句柄之和可能跟的终结符，因为这些终结符确实是规范句型中跟在句柄之后的
  - 不要看Follow集，而是要看展望符(First集)

解决的办法

- **LR(1) 分析**
  - 基本思想:
    - 对于非终极符的每个不同出现求其*后继终极符*, 称为*展望符*;

      $S' \rightarrow \delta A\omega \rightarrow \delta\alpha\beta\omega$，如果$a \in First(\omega)$，则有 $A \rightarrow \bullet\alpha\beta$, $A \rightarrow \alpha\bullet\beta$, $A \rightarrow \alpha\beta\bullet$ 的展望符a，记$(A \rightarrow \bullet\alpha\beta, a)$, $(A \rightarrow \alpha\bullet\beta, a)$, $(A \rightarrow \alpha\beta\bullet, a)$都是有效的LR(1)项目；

      如果$a \in Follow(A)$但$a \notin First(\omega)$，a不属于$A \rightarrow \bullet\alpha\beta$, $A \rightarrow \alpha\bullet\beta$, $A \rightarrow \alpha\beta\bullet$的展望符集
    - 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*展望符集*;

解决的办法

- **LR(1) 分析**
  - 分析步骤:
    - 构造 LR(1) 自动机
      - LR(1) 项目 = LR(0)项目+展望符集
    - 生成 LR(1)分析表 (action & goto)
    - LR(1)驱动程序 = LR(0)驱动程序

LR(1) 自动机

- **LR(1) 项目**
  - 两个部分： (A→ α•β, {a, …})
    - (1) LR(0) 项目: A→ α•β
    - (2) 展望符集: {a, …}，表示非终极符A此次出现的所有可能follow符号。
  - 例如：
    - 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) 自动机

- **LR(1) 项目集合*关于符号X的投影***
  - IS 是 LR(1) 项目的集合;
  - X 是一个符号;
  - $IS_{(X)}$ 表示项目集IS关于X的投影:
  - $IS_{(X)} = \{(S \rightarrow \alpha X \bullet \beta, ss) \mid (S \rightarrow \alpha \bullet X \beta, ss) \in IS, X \in V_T \cup V_N\}$

投影对展望符集没有影响！

- **IS = {(A → A•Bb, {a,b}), (B → a•, #), (B → b•B, {b})}**
- **X = B**
- **$IS_{(B)}$ = {(A → AB•b, {a, b}), (B → bB•, {b})}**

LR(1) 自动机

- **LR(1)项目集合的*闭包***
  - IS 是LR(1)项目的集合;
  - 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$\{A\rightarrow\alpha 1, \ldots, A\rightarrow\alpha n\}$

    则增加如下LR(1)项目到CLOSURE(IS)

    $\{(A\rightarrow \bullet\alpha 1, ss'), \ldots, (A\rightarrow \bullet\alpha n, ss')\}$,

    其中 $ss' = first(\pi)$, 如果符号串π不导出空;

    $ss' = (first(\pi)-\{\varepsilon\}) \cup ss$, 如果符号串$\pi$导出空;

    [3] 重复[2]直到 CLOSURE(IS)收敛;

LR(1)自动机

- **goto函数()**
  - IS 是 LR(1) 项目集;
  - X 是一个符号;
  - goto(IS, X) = CLOSURE(IS $_{(X)}$)

LR(1)自动机的构造过程

- [1]增广产生式 $Z \rightarrow S$
- [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$X \in \Sigma$,

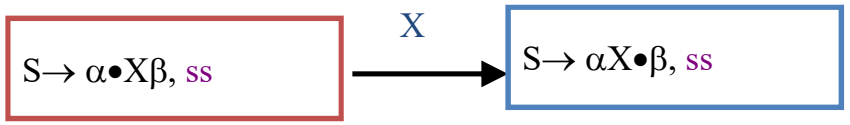
  计算IS' = goto(IS, X),

  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;

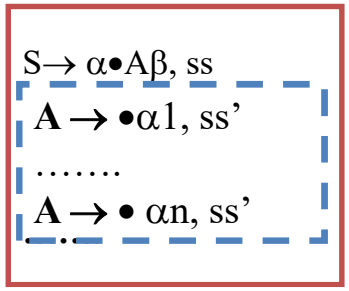
如何计算展望符集?

- **投影得到的项目**
  - **继承**



1 闭包新产生的项目

– 扩展



ss' = first(β), 如果β不导出空;

ss' = (first(β)-{ε}) ∪ss, 如果β导出空;

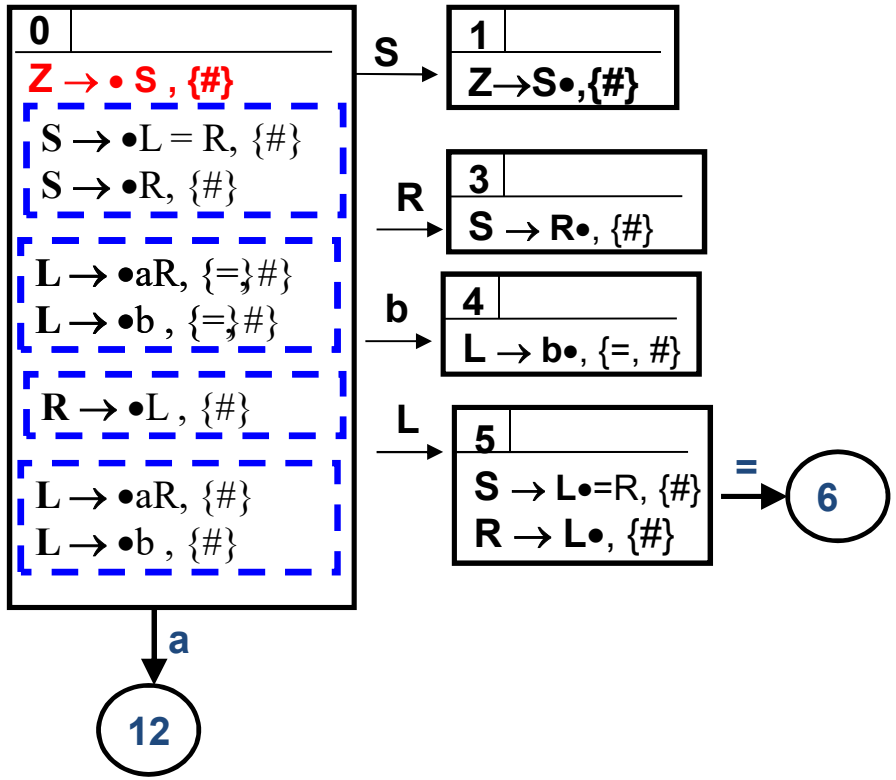
LR(1)自动机的构造

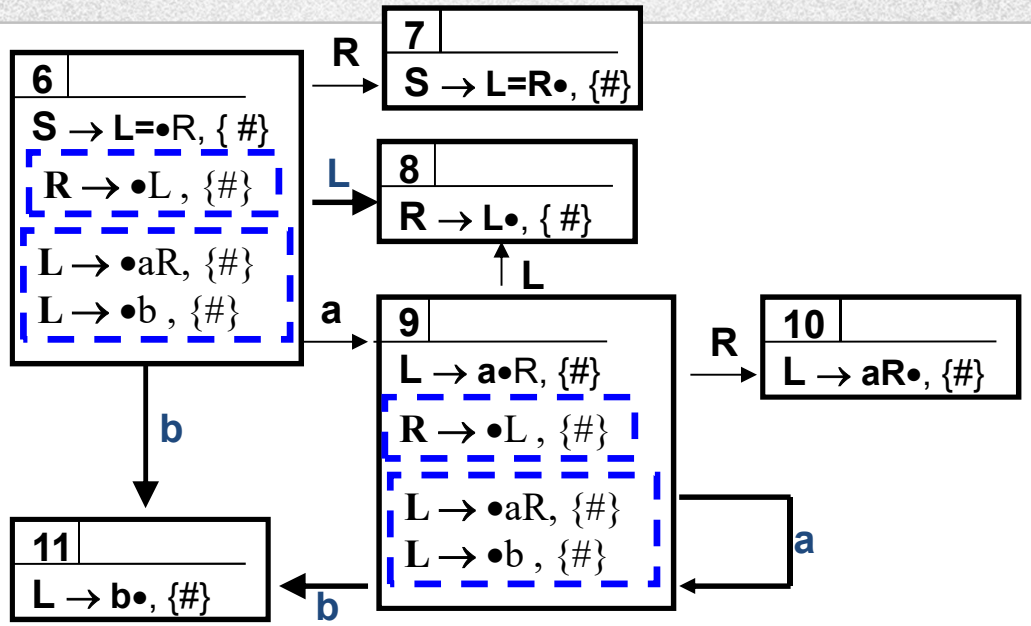
V_T = {a, b, =}

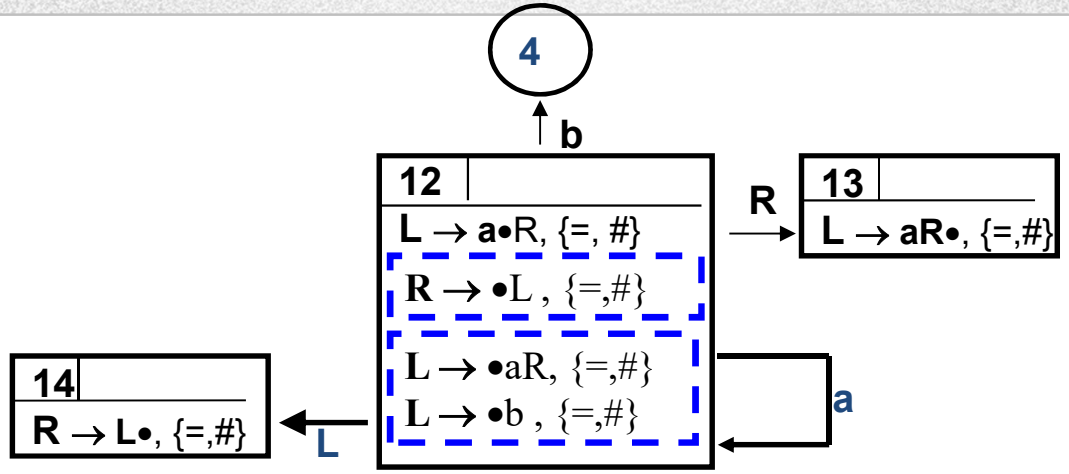
V_N = {S, L, R}

S = S

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




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
L
14
R → L•, {=,#}

LR(1) 分析表

- action表

(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

终极符 / 状 态	a_1	…	#
S_1			
…			
S_n			

LR(1) 分析表

- goto表

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

非终极符 / 状态	A_1	…	A_n
S_1			
…			
S_n			

LR(1) 分析表

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

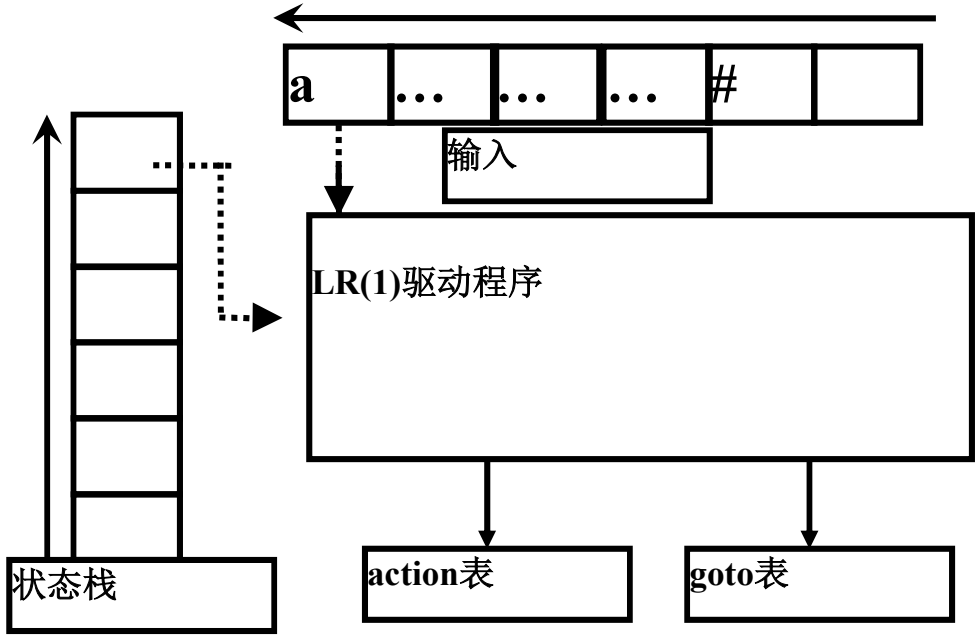
LR(1) 分析表 (接上页.)

	Action 表					Goto 表		
	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) 文法

- **给定一个上下文无关文法 *G***
  - LR_1 是文法G的 LR(1) 自动机
  - A_1 *是G的action表*
  - 如果对于任意一个状态s和任意的一个终极符a, A_1(s,a)只有一个唯一的动作, 则文法G称为*LR(1)文法*;
    - Shift
    - Reduce
    - Accept
    - Error

LR(1) 分析方法



LR(1) 分析驱动程序

- **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;

如何在LR分析时生成语法分析树？

LR(1)分析过程

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) $R \rightarrow L$
}

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

7.4 LALR(1)

LR(1)分析存在的问题

- 为消除冲突，引入太多的状态;
- 有些状态含有完全相同的LR(0)项目部分，只有展望符部分是不同的;
- LR(1)**项目的心**：如果(A→α•β, ss) 是一个LR(1)项目，则其中的LR(0)项目部分称为它的心;
- LR(1)自动机**状态的心**：一个状态所含有的所有LR(1)项目的心;
- **同心状态**：如果两个LR(1)状态具有相同的心，则称这两个状态为同心状态。

LALR(1) 分析

- **主要思想**
  - 合并文法G的LR(1)自动机中的同心状态，得到的自动机称为LALR(1)自动机；
  - 若这个得到的LALR(1)自动机没有冲突，则称文法G是LALR(1)文法。
- **LALR(1)分析过程**
  - 构造LALR(1)自动机
  - 构造LALR(1)分析表(同LR(1)分析表构造方法)
  - LALR(1)驱动程序 = LR(1)驱动程序

如何构造LALR(1)自动机

- **第一种途径：**
  - 首先构造LR(1)自动机
  - 然后合并其中的同心状态
  - 该方法简单，但不现实(not practical)!

Step1:构造LR(1)自动机

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

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

$S = S$

P:

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

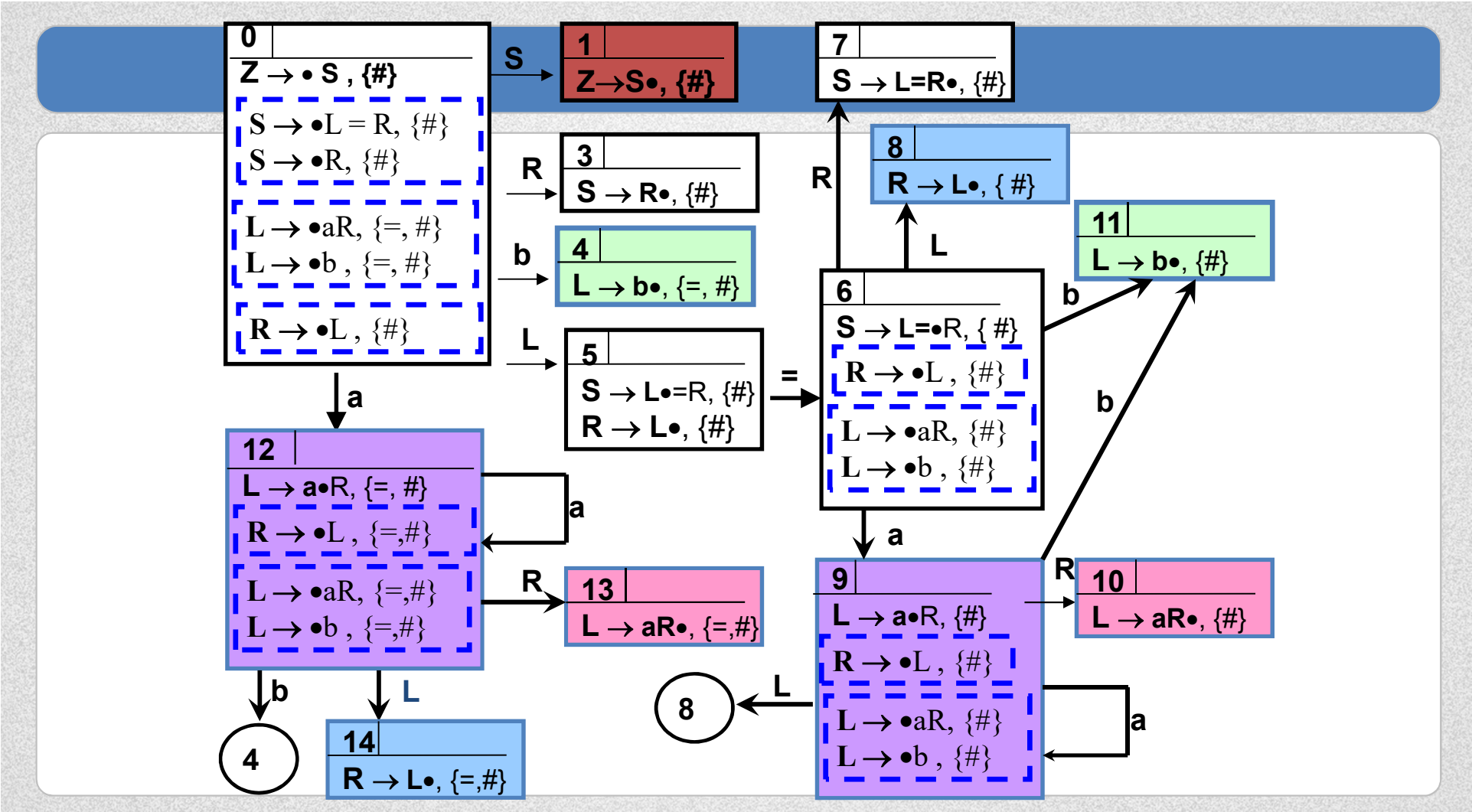
(2) $S \rightarrow R$

(3) $L \rightarrow aR$

(4) $L \rightarrow b$

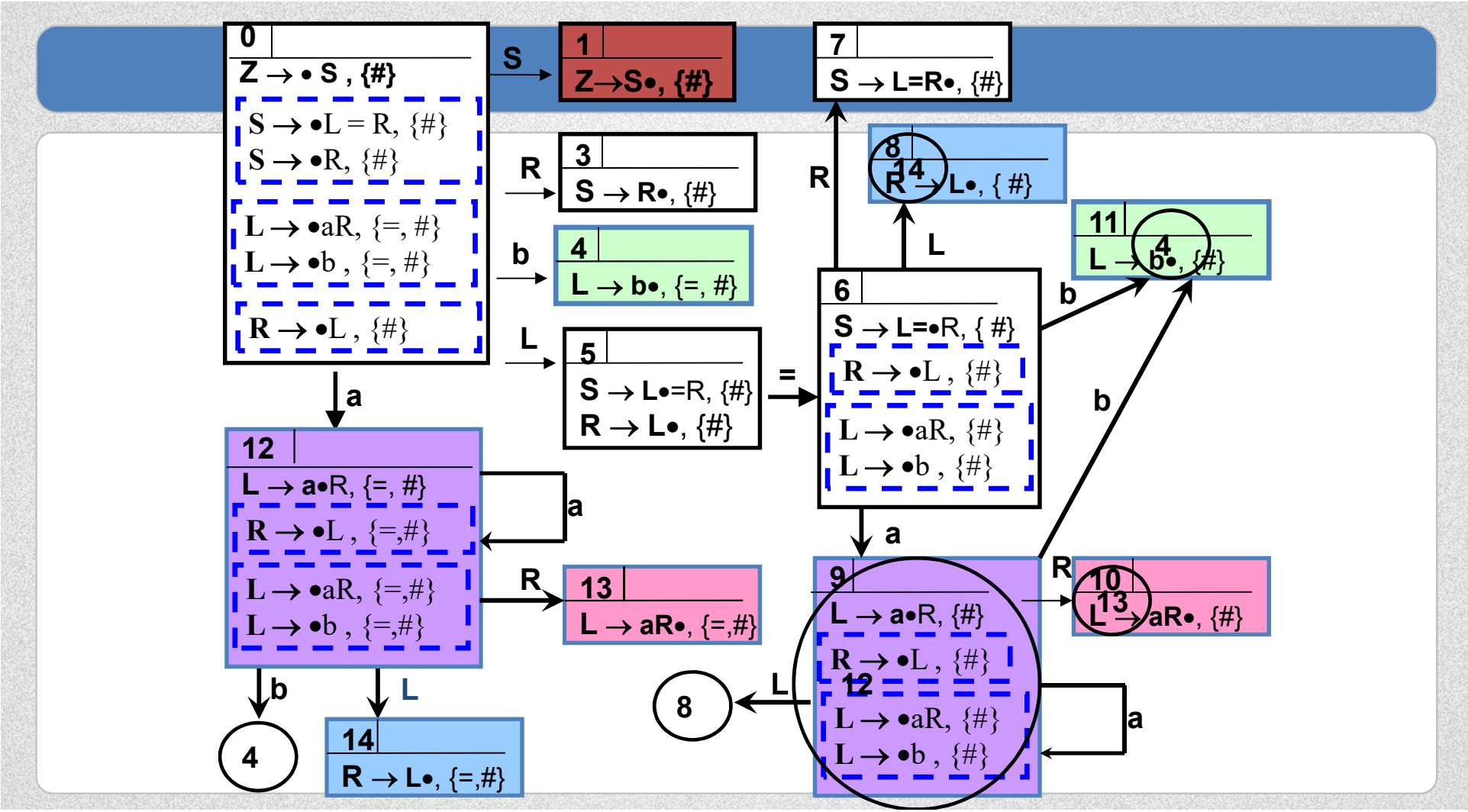
(5) $R \rightarrow L$

}


0
Z → • S , {#}
S → •L = R, {#}
S → •R, {#}
L → •aR, {=, #}
L → •b , {=, #}
R → •L , {#}
S
1
Z→S•, {#}
R
3
S → R•, {#}
b
4
L → b•, {=, #}
L
5
S → L•=R, {#}
R → L•, {#}
=
6
S → L=•R, { #}
R → •L , {#}
L → •aR, {#}
L → •b , {#}
7
S → L=R•, {#}
R
8
R → L•, { #}
L
b
11
L → b•, {#}
b
a
12
L → a•R, {=, #}
R → •L , {=,#}
L → •aR, {=,#}
L → •b , {=,#}
a
R
13
L → aR•, {=,#}
b
L
4
14
R → L•, {=,#}
a
9
L → a•R, {#}
R → •L , {#}
L → •aR, {#}
L → •b , {#}
R
10
L → aR•, {#}
L
8
a
b

Step2: 合并同心状态

- 状态4和11同心
- 状态8和14同心
- 状态10和13同心
- 状态9和12同心


0
Z → • S , {#}
S → •L = R, {#}
S → •R, {#}
L → •aR, {=, #}
L → •b , {=, #}
R → •L , {#}
S
1
Z→S•, {#}
R
3
S → R•, {#}
b
4
L → b•, {=, #}
L
5
S → L•=R, {#}
R → L•, {#}
=
6
S → L=•R, { #}
R → •L , {#}
L → •aR, {#}
L → •b , {#}
7
S → L=R•, {#}
R
8
14
R → L•, { #}
L
11
4
L → b•, {#}
b
b
a
12
L → a•R, {=, #}
R → •L , {=,#}
L → •aR, {=,#}
L → •b , {=,#}
a
R
13
L → aR•, {=,#}
a
9
L → a•R, {#}
R → •L , {#}
12
L → •aR, {#}
L → •b , {#}
R
10
13
L → aR•, {#}
b
L
8
L
b
4
L
14
R → L•, {=,#}

LR(1) 分析表

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

LR(1) 分析表 (接上页.)

	Action 表					Goto 表		
	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								

LALR(1) 分析表

	Action 表					Goto 表		
	a	b	=	#		S	L	R
0	S12	S4				1	5	3
1				Accept				
3				R2				
4			R4	R4				
5			S6	R5				
6	S12	S4					14	7
7				R1				
12	S12	S4					14	13
13			R3	R3				
14			R4	R4				

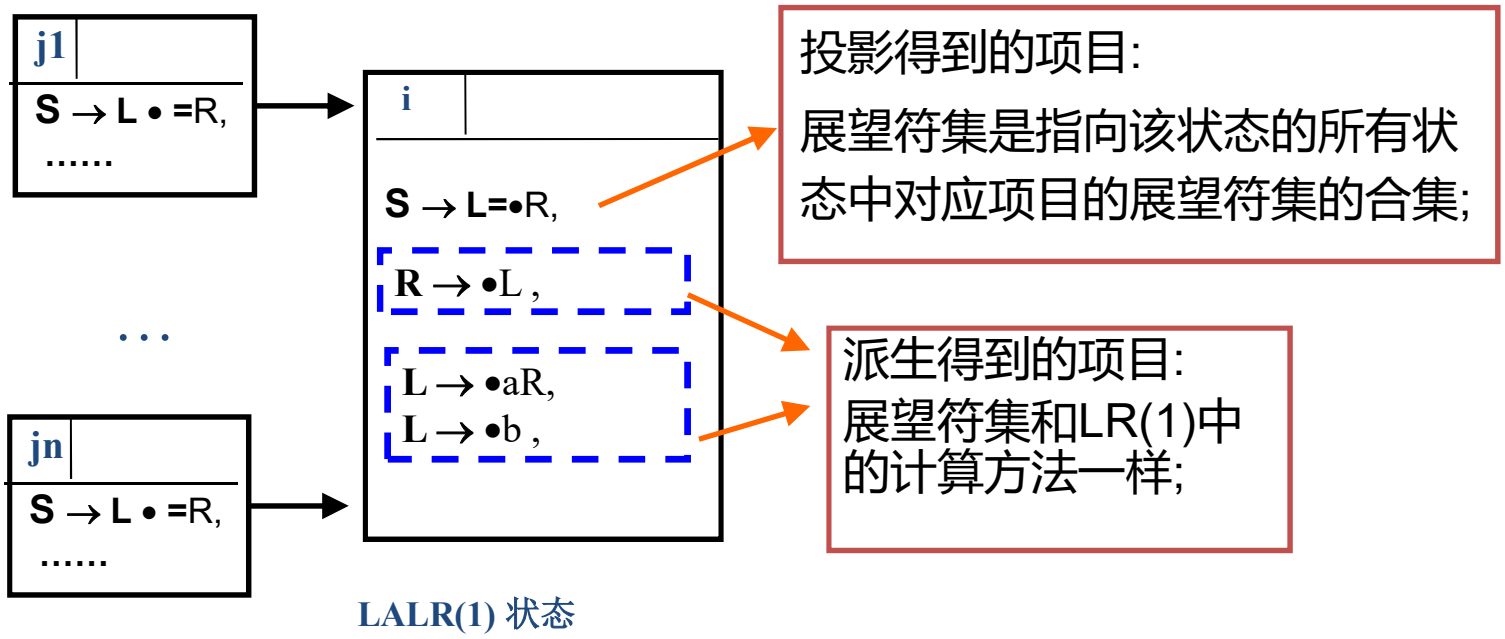
LALR(1) 自动机

- **对于给定的上下文无关文法 *G***
  - G的LALR(1)项目跟LR(1)项目形式相同;
  - LALR(1)自动机中每个状态S中各个项目的展望符集是把LR(1)自动机中所有和S同心的状态的对应项目的展望符集合合并后得到的;
  - 如果每个LALR(1)的状态都用该状态的心(LR(0)项目)替换，则LALR(1)自动机和它的LR(0)自动机相同;
  - G的LALR(1) 自动机的状态数同LR(0)自动机的状态数相同；
  - 能不能用LR(0)自动机构造LALR(1)自动机呢？

如何构造LALR(1)自动机

- **第二种途径：**
  - 首先构造LR(0)自动机
  - 然后为每个状态的每个LR(0)项目计算展望符集；
  - 是实际应用中采用的方法！
  - 关键是如何计算展望符呢？

LALR(1)展望符的计算方法



LALR(1) 状态

构造LALR(1) 自动机

V_T = {a, b, =}

V_N = {S, L, R}

S = S

P:

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

(2) $S \rightarrow R$

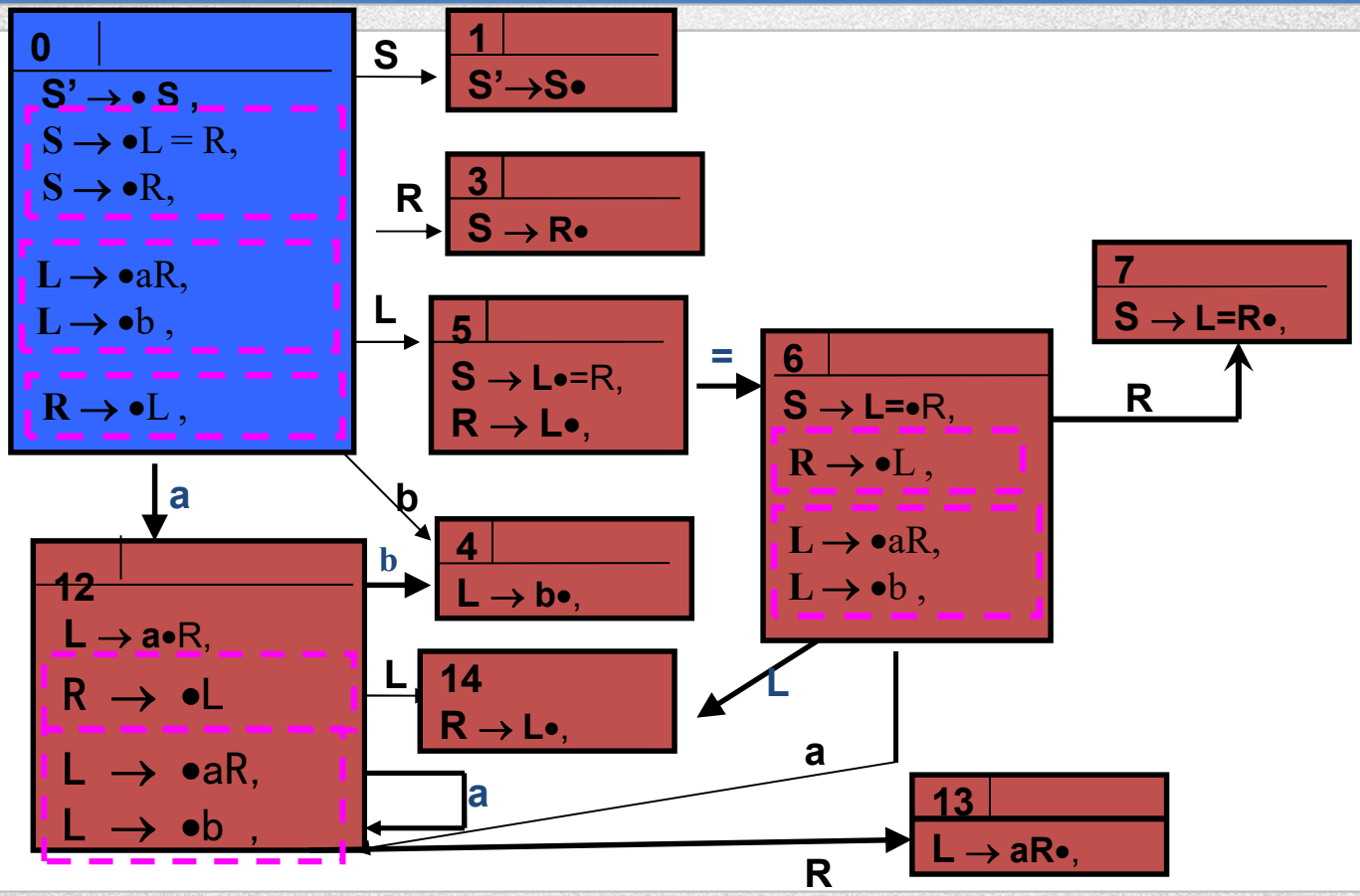
(3) $L \rightarrow aR$

(4) $L \rightarrow b$

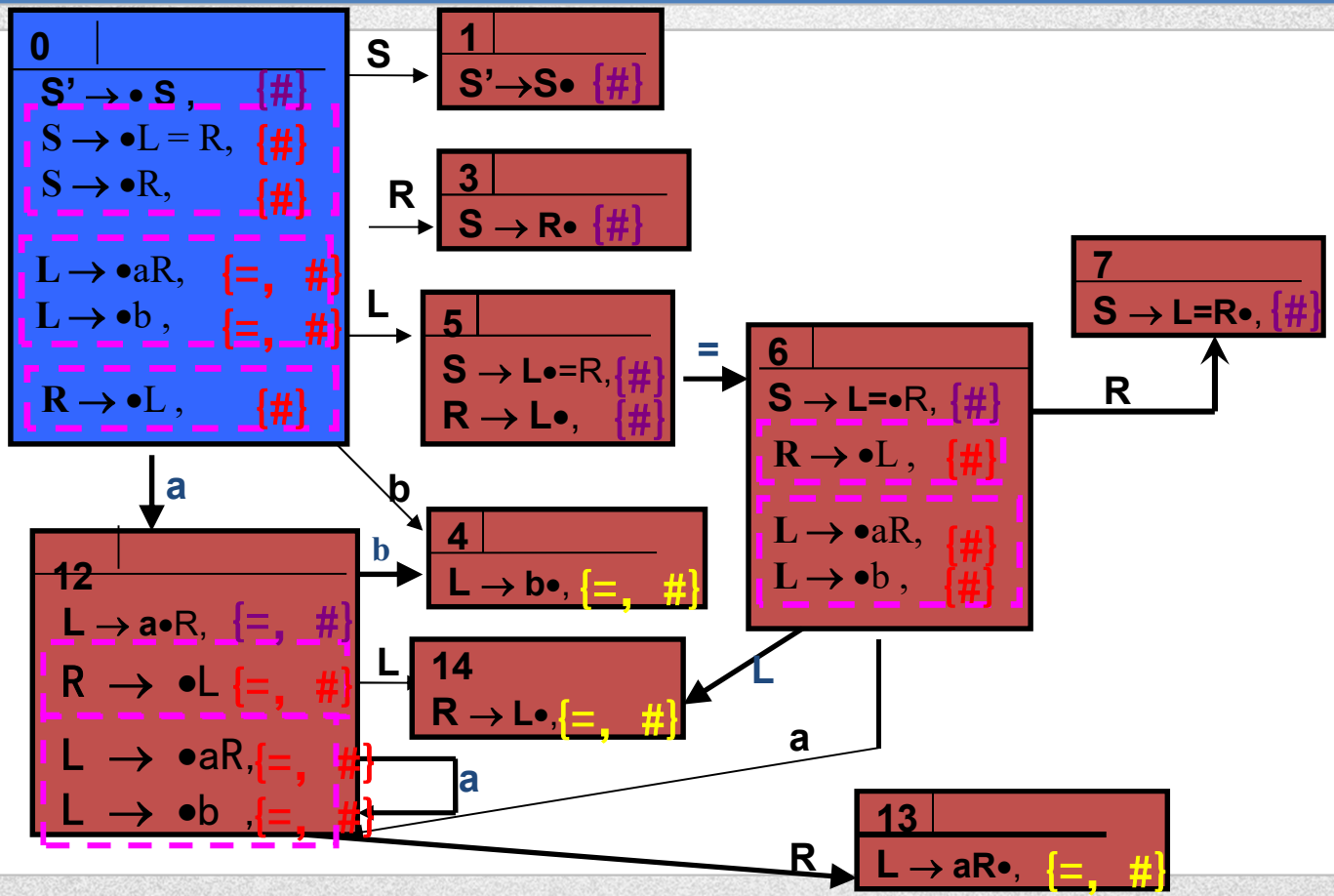
(5) $R \rightarrow L$

}

Step1: 构造LR(0) 自动机



Step2:计算展望符


0
S' → • S , {#}
S → •L = R, {#}
S → •R, {#}
L → •aR, {=, #}
L → •b , {=, #}
R → •L , {#}
1
S'→S• {#}
3
S → R• {#}
5
S → L•=R, {#}
R → L•, {#}
6
S → L=•R, {#}
R → •L , {#}
L → •aR, {#}
L → •b , {#}
7
S → L=R•, {#}
4
L → b•, {=, #}
12
L → a•R, {=, #}
R → •L {=, #}
L → •aR, {=, #}
L → •b , {=, #}
14
R → L•, {=, #}
13
L → aR•, {=, #}
S
R
L
=
R
a
b
b
L
L
a
a
R

LALR(1) 文法

- **如果文法G的LALR(1) 自动机中没有冲突,则文法G称为*LALR(1) 文法*;**
  - 注意: 由于对LR(1)自动机中的同心状态进行了合并, 把展望符集进行合并, 因此有可能导致冲突, 从而不是LALR(1)文法;
- **合并同心状态只能产生归约-归约冲突，不会产生移入-归约冲突，为什么？**

LALR(1) 文法

- **合并Action表**
  - 出错与出错合并：结果仍为出错，无冲突；
  - 移进与移进合并；
  - 出错与移进合并：不会出现此类情况，因为出错项目与移进项目不同心
  - 移进与规约合并：不会出现此类情况，因为不可能不同心，例如 I1：L→ i• #|= ；I2：L→ i•a xxx

LALR(1) 文法

- **合并Action表**
  - 规约与出错合并：规定它做规约
    - 由此可见，LALR与LR(1)相比，放松了报错条件，但由于移进预测能力没有减弱，所以在下一个符号进栈前总能报错，所以对错误的定向能力没有减弱
  - 规约与规约合并：
    - 按同一个产生式规约，无冲突；
    - 按不同产生式规约，将造成冲突，因此LALR能力弱于LR(1)

综上所述，可得：只要合并同心之后，不存在按不同产生式的规约-规约冲突，由LR(1)项目集族总能构造成LALR分析表

LR(0), SLR(1), LR(1)和LALR(1)比较

- **状态数:**
  - LR(1) > LALR(1) = SLR(1) = LR(0)
- **展望符的确定:**
  - LR(0)没有展望符；
  - SLR(1)取follow集；
  - LR(1)取不同位置的follow集；
  - LALR(1)取同心项的展望符的并集；
- **向前看输入符:**
  - SLR(1),LR(1)和LALR(1)向前看一个输入符；
  - LR(0)不看;
- **分析能力: LR(1) ⊃ LALR(1) ⊃ SLR(1) ⊃ LR(0)**

作业

- 教材P164: 4.6.2，4.6.3，4.6.5，4.6.6
- 教材P177: 4.7.3，4.7.4，4.7.5

Thank you!