Bottom-Up Parsing

1

Bottom-Up Parsing

1. **Overview**
2. **LR(0) Parsing and SLR(1) Parsing**
3. **LR(1) Parsing and LALR(1) Parsing**

2

overview

- ***Bottom-up parsing* traces out a rightmost derivation of the input string, but the steps of the derivation occur in reverse order.**
- **The most general bottom-up algorithm is LR(k) parsing.**
  - **LR(0) parsing**
  - **SLR(1) parsing**
  - **LR(1) parsing**
  - **LALR(1) parsing**

3

How to Implement

- **A bottom-up parser uses an explicit stack to perform a parse.**
  - **contains tokens, nonterminals, some extra state information**
  - **Initial: empty**
  - **Success: start symbol**
- **A schematic for bottom-up parsing:**

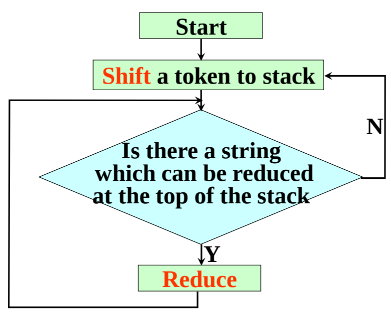
Stack	InputString	Actions
$	*InputString*$	
……	……	……
$StartSymbol	$	accept

4

Two Actions

- **Two possible actions (besides "accept"):**
  - ✓ ***Shift* a terminal from the front of the input to the top of the stack; 移进/移入**
  - ✓ ***Reduce* a string α at the top of the stack to a nonterminal A, given the production A→α;归约**
- **Shift-reduce parser**
- **Grammars are always augmented with a new start symbol**
  增广文法 $S' \rightarrow S$

5



6

The LR Parsing Algorithm

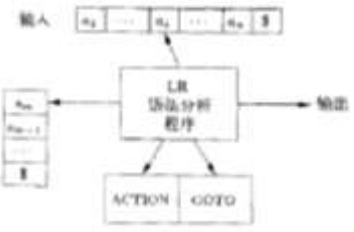


图 4-35 一个 LR 语法分析器的模型

7

The LR Parsing Algorithm

LR分析过程：

假设i是当前栈顶状态，a是下一个输入符号，分析过程如下：

- 初始：符号$和状态$s_0$进栈
- 用状态i和输入符号a查表，分别执行以下动作：
  - **Shift移进**：若ACTION[i, a]=**sj**，则将符号a和状态j进栈（实际上，a可以不用进栈）
  - **Accept接受**：若ACTION[i, a]=acc，则表示语法正确，分析成功完成

8

The LR Parsing Algorithm

- **Reduce规约**：若ACTION[i, a]=**r**(A→γ)，则执行规约操作：
  ① 弹出栈顶|γ|个符号和|γ|个状态，得到目前栈顶状态k
  ② 若m=GOTO[k, A]，将状态m和符号A压入分析栈（实际上，A可以不用进栈）
- **Error出错**：若ACTION[i, a]=出错，则执行错误处理。

9

LR语法分析伪代码

输入串为**w$**，符号$和初始状态s_0压入分析栈：

```
令 a 为 w$ 的第一个符号；
while(1) { /* 永远重复 */
    令 s 是栈顶的状态；
    if ( ACTION[s, a] = 移入 t ) {
        将 t 压入栈中；
        令 a 为下一个输入符号；
    } else if ( ACTION[s, a] = 归约 A → β ) {
        从栈中弹出|β|个符号；
        令 t 为当前的栈顶状态；
        将 GOTO[t, A] 压入栈中；
        输出产生式 A → β；
    } else if ( ACTION[s, a] = 接受 ) break; /* 语法分析完成 */
    else 调用错误恢复例程；
}
```

10

Bottom-Up Parsing

1. **Overview**
2. **LR(0) Parsing and SLR(1) Parsing**
3. **LR(1) Parsing and LALR(1) Parsing**

11

2. LR(0) Parsing and SLR(1) Parsing

(1) LR(0) ITEMS

(2) LR(0) Parsing

(3) SLR(1) Parsing

12

(1) LR(0) ITEMS

- **An LR(0) item:**
  - **A production choice with a distinguished position in its right-hand side**
  - **Indicating the distinguished position by a period**
- **Example:**
  - **A→βγ is a production choice.**
  - **A→β·γ is an LR(0) item.**
- **These are called LR(0) items because they contain no explicit reference to lookahead**

13

(1) LR(0) ITEMS

Example: The grammar:

$S' \to S$
$S \to (S)S$
$S \to \varepsilon$

This grammar has eight items:

$S' \to \cdot S$	$S' \to S\cdot$	$S \to \cdot(S)S$
$S \to (\cdot S)S$	$S \to (S\cdot)S$	$S \to (S)\cdot S$
$S \to (S)S\cdot$	$\boxed{S \to \cdot}$	

14

(1) LR(0) ITEMS

- **An item records an intermediate step in the recognition of the right-hand side of a rule choice;**
- **A→β·γ means that β has already been seen, it may be possible to derive the next input tokens from γ;**
- **A→·α initial item** 初始项目
- **A→α· complete item** 完整项目

15

(2) LR(0) Parsing

项目集的闭包(CLOSURE)

定义：若I是文法G的一个项目集，则CLOSURE(I)包括如下项目：

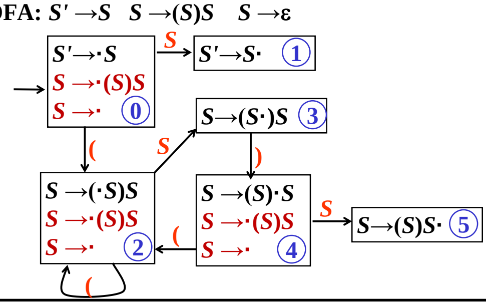
- **Kernal items:** I中所有项目都在CLOSURE(I)中
- **Closure items:** 若A→α·Bβ (B∊N)在I中，则将B的所有产生式对应的初始项目（如: B→·γ_1, B→·γ_2）都加入到CLOSURE(I)中（实际上就是通过ε-闭包加入的项目）

例：文法	若项目集I={[*E'*→·*E*]}，则CLOSURE(I)
E'→*E*	={[*E'*→·*E*],
E →*E*+*T* |*T*	[*E*→·*E*+*T*], [*E*→·*T*],
T →*T***F* |*F*	[*T*→·*T**F], [*T*→·*F*],
F →(*E*) | id	[*F*→·(*E*)], [*F*→·id]}

16

(2) LR(0) Parsing

Example DFA: $S' \to S \quad S \to (S)S \quad S \to \varepsilon$



17

(2) LR(0) Parsing

状态间的转换 **GOTO函数**

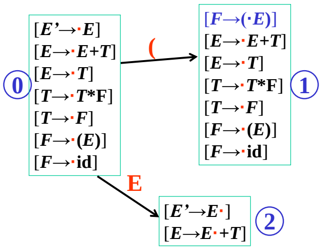
- **The transitions of items:**
  - **Consider the item**
    A→α·Xη and A→αX·η (X∊(N∪T))
  - **There has a transition on the symbol X from the first item to the second item.**
- **GOTO(I, X):** 项目集I中的项目，在输入符号X后，得到的项目集的闭包.

18

(2) LR(0) Parsing

- Graphical form of GOTO(I, X):

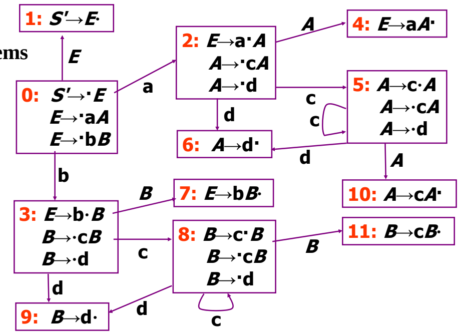
GOTO(I, X)：项目集I中的项目，在输入符号X后，得到的项目集的闭包.



19

DFA of LR(0) items

S'→*E*
E→a*A* | b*B*
A→c*A* | d
B→c*B* | d



20

(2) LR(0) Parsing

构造LR(0)分析表：

① 构造增广文法的LR(0)项目集族的DFA

② 根据各状态中的项目填写分析表：

- 若从状态i到j有一个基于符号X的转换
  - ✓ X∈T时，则Action[i, X]=sj;
  - ✓ X∈N时，则Goto[i, X]=j;
- 若状态i中有完整项目[A→γ·]，且A≠S'，则对于所有X∈T，Action[i, X]=r(A→γ);
- 若状态i中有完整项目[S'→S·]，则Action[i, $]=acc;

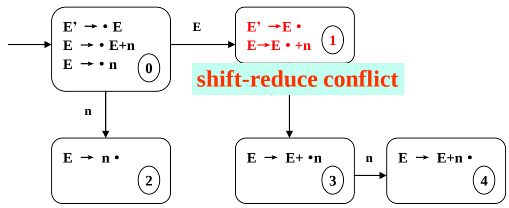
21

Why we need SLR(1) parsing?

- **The grammar**

  E'* →*E

  ***E* →*E*+n | n**



22

(3) SLR(1) Parsing

构造SLR(1)分析表：

① 构造增广文法的LR(0)项目集族的DFA

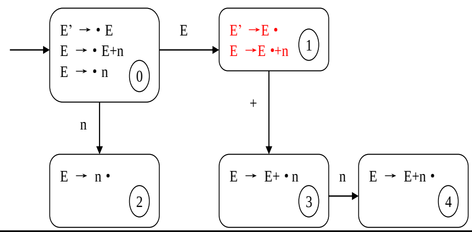
② 根据各状态中的项目填写分析表：

- 若从状态i到j有一个基于符号X的转换
  - ✓ X∈T时，则Action[i, X]=sj;
  - ✓ X∈N时，则Goto[i, X]=j;
- 若状态i中有完整项目[A→γ·]，且A≠S'，则对于所有a∈Follow(A)，Action[i, a]=r(A→γ);
- 若状态i中有完整项目[S'→S·]，则Action[i, $]=acc;

23

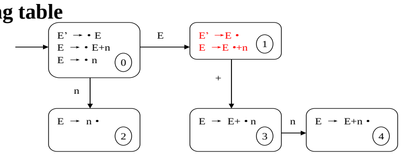
- **Example：Consider the grammar**

  E'→E　E→E + n | n
- Construct the SLR(1) parsing table
  - Follow(E') ={$}，Follow(E) = { $, +}



24

The SLR(1) parsing table



State	Input			Goto
	n	+	$	E
0	S2			1
1		S3	accept	
2		r(*E*→*n*)	r(*E*→*n*)	
3	S4			
4		r(*E*→*E*+*n*)	r(*E*→*E*+*n*)	

Follow(E') ={$}
Follow(E) = {$, +}

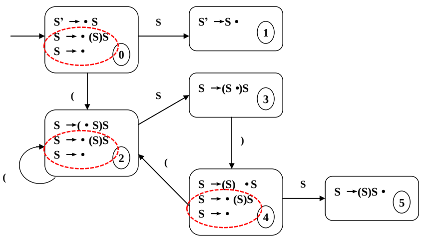
25

(3) SLR(1) Parsing

- **A grammar is SLR(1) if and only if, for any state s, the following two conditions are satisfied:**
  - **For any item *A*→α·Xβ(X∈T) in s, there is no complete item *B*→γ· in s with X∈Follow(*B*).**
  - **For any two complete items *A* →α· and *B*→β· in s, Follow(*A*)∩Follow(*B*) is empty.**
- **Violate the first condition: shift-reduce conflict**
- **Violate the second condition: reduce-reduce conflict**

26

Example S'→S S → (S)S|*ε*
Follow sets: Follow(S') = {$}, Follow(S)={$,) }



Is this grammar an SLR(1) grammar?

27

课堂练习

Consider the grammar:

S → SS+ | SS* | a

(a) Construct the DFA of LR(0) items for this grammar;

(b) Construct the SLR(1) parsing table.

(c) Is this grammar an SLR(1) grammar? Give the reason.

(d) Show the parsing stack and actions of an SLR(1) parser, given the input string **aa*a+** .

28

Bottom-Up Parsing

1. **Overview**
2. **LR(0) Parsing and SLR(1) Parsing**
3. **LR(1) Parsing and LALR(1) Parsing**

29

3. General LR(1) and LALR(1) Parsing

(1) Finite Automata of LR(1) Items

(2) LR(1) parsing table

(3) LALR(1) parsing

30

(1) Finite Automata of LR(1) Items

- The SLR(1) method:
  - Applies lookaheads after the construction of the DFA of LR(0) items
  - The **construction of DFA ignores lookaheads**
- The general LR(1) method:
  - **Using a new DFA with the lookaheads built into its construction,** LR(1) items include a single lookahead token in each item.
  - An LR(1) item is a pair:

    [an LR(0) item, a lookahead token]

  LR(1) item example: [A→α·β, a]

31

(1) Finite Automata of LR(1) Items

- **LR(1)分析中CLOSURE(I)的定义：**
  - **All items in I are also in CLOSURE(I);**
  - **If an item [A→α·Bγ, a] B∈N is in I, items [B→·β, b] for every B→β and *every token b*∈First(γa) are in CLOSURE(I).**

32

(1) Finite Automata of LR(1) Items

- **Start state: CLOSURE({[*S'*→·*S*, $]})**
- Each CLOSURE(I) is a state of DFA

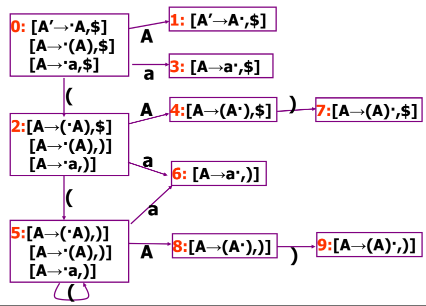
  [A→α·Bγ, a]
  [B→·β, b]

- GOTO: The transitions between states
  - Similar to LR(0) transitions except keeping track of lookaheads

$[A\to\alpha\cdot X\gamma,\ a] \xrightarrow{X} [A\to\alpha X\cdot\gamma, a]$

33

Example G: A→(A) | a **The DFA of sets of LR(1) items**



34

(2) LR(1) parsing table

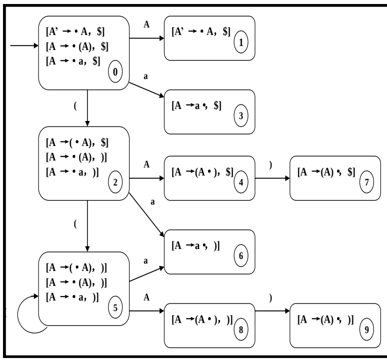
构造LR(1)分析表：

① 构造增广文法的LR(1)项目集族的DFA

② 根据各状态中的项目填写分析表：

- 若从状态i到j有一个基于符号X的转换
  - ✓ X∈T时，则Action[i, X]=sj;
  - ✓ X∈N时，则Goto[i, X]=j;
- 若状态i中有完整项目[A→γ·, a]，且A≠S'，则Action[i, a]=r(A→γ)；
- 若状态i中有完整项目[S'→S·, $]，则Action[i, $]=acc；

35



(l) A→(A) (2) A→a

S	Input				Goto
	(	a	)	$	A
0	s2	s3			1
1				acc	
2	s5	s6			4
3				r2	
4			s7		
5	s5	s6			8
6			r2		
7				r1	
8			s9		
9			r1		

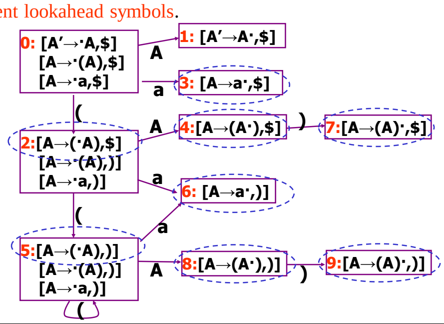
36

(2) LR(1) parsing table

- A grammar is LR(1) if and only if, for any state s, the following two conditions are satisfied.
  1. For any item [A→α·Xβ, a](X∈T) in **s**, there is no item in **s** of the form [B→γ·, X] (otherwise there is a shift-reduce conflict).
  2. There are no two items in **s** of the form [A→α·,a] and [B→β·, a] (otherwise, there is a reduce-reduce conflict).

37

- In the DFA of sets of LR(1) items, many states have the same LR(0) items, and different lookahead symbols.



38

(3) LALR(1) parsing

- The LALR(1) parsing algorithm:
  - Identify all such states and combine their lookaheads;
  - Then, we have a DFA identical to the DFA of LR(0) items, except that each state consists of items with sets of lookaheads.
- In the case of complete items these lookahead sets are often smaller than the corresponding Follow sets.

39

(3) LALR(1) parsing

- Constructing the DFA of LALR(l) items:
  - Constructed from the DFA of LR(l) items by identifying all states that have the same core
  - And forming the union of the lookahead symbols for each LR(0) item
- Each LALR(l) item in this DFA will have an LR(0) item as its first component and a set of lookahead tokens as its second component.
- 用LALR(1)项目的DFA构造分析表的方法与用LR(1)项目的DFA构造分析表完全一样。

40

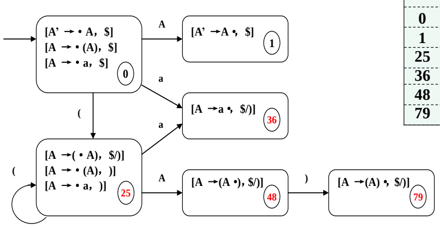
(3) LALR(1) parsing

- **Example**

  (l) A→(A)

  (2) A→**a**

State	Input				Goto
	(	a	)	$	A
0	s25	s36			1
1				acc	
25	s25	s36			48
36			r2	r2	
48			s79		
79			r1	r1	



41

(3) LALR(1) parsing

- It is possible for the LALR(l) parsing table construction to create parsing conflicts that do not exist in general LR(1) parsing table, but this rarely happens in practice
- Indeed, if a grammar is LR(l), then the LALR(l) parsing table cannot have any shift-reduce conflicts, there may be reduce-reduce conflicts.

42

(3) LALR(1) parsing

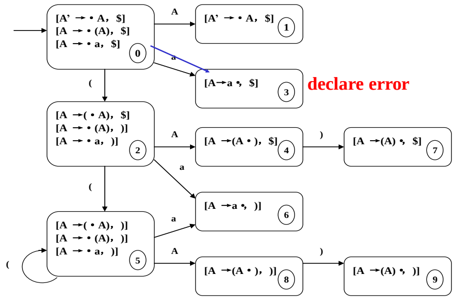
- If a grammar is SLR(l), then it is LALR(l)
- LALR(1) parsers often do as well as general LR(1) parsers in removing typical conflicts that occur in SLR(l) parsing
- If the grammar is already LALR(l), the only consequence of using LALR(l) parsing over general LR parsing is that, in the presence of errors, some spurious reductions may be made before error is declared
- For example:

  Given the erroneous input string **a)**

43

The DFA of LR(1)

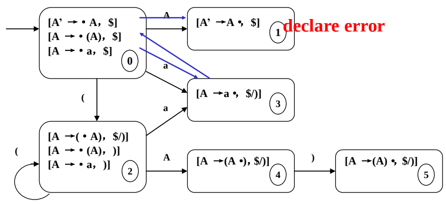
input string **a)**



44

The DFA of LALR(1)

input string **a)**



Note:

➢**LALR(l) parser: (1) reduce(A→a) (2)declare error**

➢**A general LR(l) parser: declare error immediately after shift a**

45