SNS COLLEGE OF TECHNOLOGY

Coimbatore-36.

An Autonomous Institution

Accredited by NBA –AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 19CSB301&Automata Theory and Compiler Design

III YEAR/ V SEMESTER

UNIT –III SYNTAX ANALYSIS AND SEMANTIC ANALYSIS

Topic: SLR Parser

Dr.B.Vinodhini
Associate Professor
Department of Computer Science and Engineering

Constructing SLR Parsing Table - Steps

To perform SLR parsing, take grammar as input and do the following:

1. Find LR(0) items.
2. Completing the closure.
3. Compute *goto*(I,X), where, I is set of items and X is grammar symbol.

LR(0) items:

An *LR(0) item* of a grammar G is a production of G with a dot at some position of the right side. For example, production A → XYZ yields the four items :

A → . XYZ
A → X . YZ
A → XY . Z
A → XYZ .

Constructing SLR Parsing Table - Steps

Closure operation:

If I is a set of items for a grammar G, then closure(I) is the set of items constructed from I by the two rules:

1. Initially, every item in I is added to closure(I).
2. If $A \rightarrow \alpha . B\beta$ is in closure(I) and $B \rightarrow \gamma$ is a production, then add the item $B \rightarrow . \gamma$ to I , if it is not already there. We apply this rule until no more new items can be added to closure(I).

Goto operation:

Goto(I, X) is defined to be the closure of the set of all items $[A \rightarrow \alpha X . \beta]$ such that $[A \rightarrow \alpha . X\beta]$ is in I.

Steps to construct SLR parsing table for grammar G are:

1. Augment G and produce G'
2. Construct the canonical collection of set of items C for G'
3. Construct the parsing action function *action* and *goto* using the following algorithm that requires FOLLOW(A) for each non-terminal of grammar.

Constructing SLR Parsing Table - Algorithm

Input : An augmented grammar G'
Output : The SLR parsing table functions *action* and *goto* for G'
Method :

1. Construct $C = \{I_0, I_1, \ldots I_n\}$, the collection of sets of LR(0) items for G'.
2. State i is constructed from I_i. The parsing functions for state i are determined as follows:
   (a) If $[A \rightarrow \alpha \cdot a\beta]$ is in I_i and $goto(I_i, a) = I_j$, then set *action*$[i,a]$ to "shift j". Here a must be terminal.
   (b) If $[A \rightarrow \alpha \cdot]$ is in I_i, then set *action*$[i,a]$ to "reduce $A \rightarrow \alpha$" for all a in FOLLOW(A).
   (c) If $[S' \rightarrow S.]$ is in I_i, then set *action*$[i,\$]$ to "accept".
3. The *goto* transitions for state i are constructed for all non-terminals A using the rule: If $goto(I_i, A) = I_j$, then $goto[i,A] = j$.
4. All entries not defined by rules (2) and (3) are made "error"
5. The initial state of the parser is the one constructed from the set of items containing $[S' \rightarrow .S]$.

Constructing SLR Parsing Table - Example

Example for SLR parsing:

Construct SLR parsing for the following grammar :

G : E → E + T | T
T → T * F | F
F → (E) | id

The given grammar is :

G : E → E + T ------ (1)
E → T ------ (2)
T → T * F ------ (3)
T → F ------ (4)
F → (E) ------ (5)
F → id ------ (6)

Step 1 : Convert given grammar into augmented grammar.

Augmented grammar :

E' → E
E → E + T
E → T
T → T * F
T → F
F → (E)
F → id

Step 2 : Find LR (0) items.

I_0 : E' → . E
E → . E + T
E → . T
T → . T * F
T → . F
F → . (E)
F → . id

Constructing SLR Parsing Table - Example

GOTO (I_4 , id)
I_5 : F → id .

GOTO (I_7 , ( )
I_4 : F → (. E)
E → . E + T
E → . T
T → . T * F
T → . F
F → . (E)
F → . id

GOTO (I_9 , *)
I_7 : T → T * . F
F → . (E)
F → . id

GOTO (I_4 , ( )
I_4 : F → (. E)
E → . E + T
E → . T
T → . T * F
T → . F
F → . (E)
F → id

GOTO (I_7 , id)
I_5 : F → id .

I_4 : F → (. E)

GOTO (I_8 ,))
I_{11} : F → (E) .

GOTO (I_4 , E)
I_8 : F → (E .)
E → E . + T

GOTO (I_6 , id)
I_5 : F → id .

GOTO (I_8 , +)
I_6 : E → E + . T
T → . T * F
T → . F
F → . (E)
F → . id

GOTO (I_4 , T)
I_2 : E → T .
T → T . * F

GOTO (I_7 , F)
I_{10} : T → T * F .

GOTO (I_4 , F)
I_3 : T → F .

Constructing SLR Parsing Table - Example

FOLLOW (E) = { $,) , +)
FOLLOW (T) = { $, + ,) , * }
FOOLOW (F) = { * , + ,) , $ }

SLR parsing table:

	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
I_0	s5			s4			1	2	3
I_1		s6				ACC			
I_2		r2	s7		r2	r2			
I_3		r4	r4		r4	r4			
I_4	s5			s4			8	2	3
I_5		r6	r6		r6	r6			
I_6	s5			s4				9	3
I_7	s5			s4					10
I_8		s6			s11				
I_9		r1	s7		r1	r1			
I_{10}		r3	r3		r3	r3			
I_{11}		r5	r5		r5	r5			

Constructing SLR Parsing Table - Example

Stack implementation:

Check whether the input **id + id * id** is valid or not.

STACK	INPUT	ACTION
0	id + id * id $	GOTO (I_0 , id) = s5 ; **shift**
0 id 5	+ id * id $	GOTO (I_5 , +) = r6 ; **reduce** by F→id
0 F 3	+ id * id $	GOTO (I_0 , F) = 3 GOTO (I_3 , +) = r4 ; **reduce** by T → F
0 T 2	+ id * id $	GOTO (I_0 , T) = 2 GOTO (I_2 , +) = r2 ; **reduce** by E → T
0 E 1	+ id * id $	GOTO (I_0 , E) = 1 GOTO (I_1 , +) = s6 ; **shift**
0 E 1 + 6	id * id $	GOTO (I_6 , id) = s5 ; **shift**
0 E 1 + 6 id 5	* id $	GOTO (I_5 , *) = r6 ; **reduce** by F → id
0 E 1 + 6 F 3	* id $	GOTO (I_6 , F) = 3 GOTO (I_3 , *) = r4 ; **reduce** by T → F
0 E 1 + 6 T 9	* id $	GOTO (I_6 , T) = 9 GOTO (I_9 , *) = s7 ; **shift**

Constructing SLR Parsing Table - Example

0 E 1 + 6 T 9 * 7	id $	GOTO (I_7 , id) = s5 ; **shift**
0 E 1 + 6 T 9 * 7 id 5	$	GOTO (I_5 , $) = r6 ; **reduce** by F → id
0 E 1 + 6 T 9 * 7 F 10	$	GOTO (I_7 , F) = 10 GOTO (I_{10} , $) = r3 ; **reduce** by T → T * F
0 E 1 + 6 T 9	$	GOTO (I_6 , T) = 9 GOTO (I_9 , $) = r1 ; **reduce** by E → E + T
0 E 1	$	GOTO (I_0 , E) = 1 GOTO (I_1 , $) = **accept**

SLR [Follow of Final positon (Reduce)]

Example: (a)

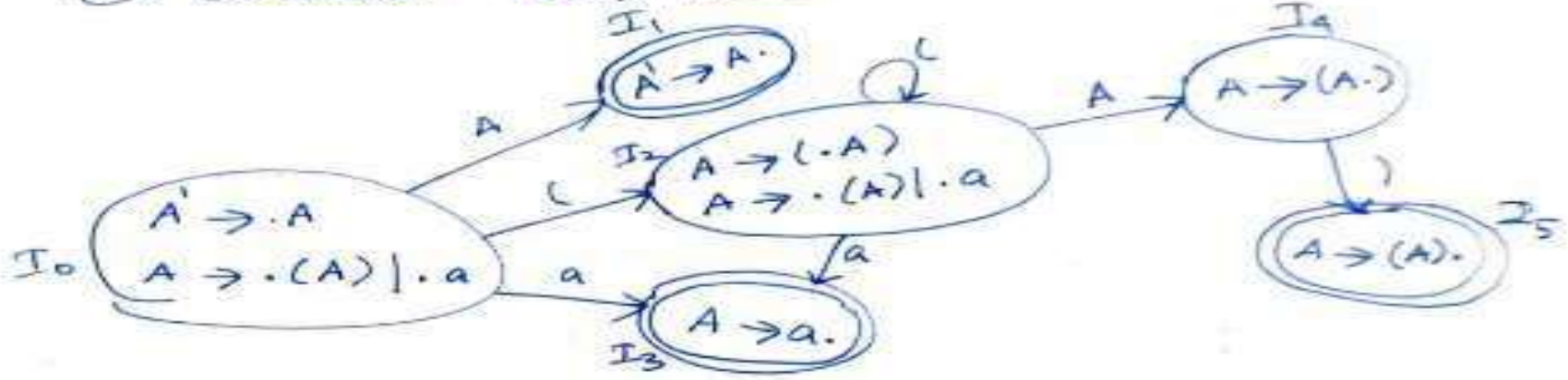
	FIRST	FOLLOW
$A \rightarrow (A) \mid a$	$\{(, a\}$	$\{\$,)\}$

① Augumented Grammar

$A' \rightarrow A$

$A \rightarrow (A) \mid a$

② Canonical LR(0) items



1. $A \rightarrow (A)$

2. $A \rightarrow a$

③ LR(0) parse Table

	ACTION				GOTO
	a	(	)	$	A
I_0	S_3	S_2			1
I_1				Acc	
I_2	S_3	S_2			4
I_3	r_2	r_2	r_2	r_2	
I_4			S_5		
I_5	r_1	r_1	r_1	r_1	

processing Input (a)

STACK	I/P	ACTION
$ 0	(a) $	
$ 0 (2	a) $	shift (2
$ 0 (2 a 3	) $	shift a 3
$ 0 (2 A 4	) $	Reduce 2
$ 0 (2 A 4) 5	$	shift) 5
$ 0 A 1	$	Reduce 1
		Accept

SLR Table

Final Items - Reduce operation (Follow of ~~(First of LHS)~~ LHS)

$A \rightarrow (A) \mid a$, FIRST $\{(, a\}$ | FOLLOW $\{), \$\}$

ITEM	ACTION				GOTO
	a	(	)	$	A
I_0	S_3	S_2			1
I_1				Accept	
I_2	S_3	S_2			4
I_3			r_2	r_2	
I_4			S_5		
I_5			r_1	r_1	


Thank
you