Bottom Up Parsers

1

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Bottom Up Parsing

- They are powerful compared with TD parsers.
- Understandably more complex.
- Left recursion is not a problem for BU parsers.
- Right recursion is a bit of problem but not serious ?
- *Not suitable for hand coding.*

2

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Bottom-up Parsing:

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

3

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Bottom Up Parsing

- Basic operation is to *shift* terminals from the input to the stack until the right-hand side of an appropriate grammar rule is seen, and then to *reduce* the stuff on the stack that matches the right-hand side to the single non-terminal of the rule. Hence, bottom-up parsers are often called *shift-reduce parsers.*
- Stack can be viewed as containing both terminals and non-terminals.
- Table-driven using an explicit stack.

4

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Example Grammar 1

$E' \rightarrow E$

$E \rightarrow E + n \mid n$

Input: 2 + 3, or $n + n$

Parse: ($ is EOF in input, also bottom of stack)

	Parsing stack	Input	Action
1	$	n + n $	shift
2	$ n	+ n $	reduce $E \rightarrow n$
3	$ E	+ n $	shift
4	$ E +	n $	shift
5	$ E + n	$	reduce $E \rightarrow E + n$
6	$ E	$	reduce $E' \rightarrow E$
7	$ E'	$	Accept

5

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Example Grammar 2

$S' \rightarrow S$

$S \rightarrow (S)\ S \mid \epsilon$

Input String: ( )

	Parsing Stack	Input	Action
1	$	( ) $	shift
2	$ (	) $	reduce $S \rightarrow \epsilon$
3	$ (S	) $	shift
4	$ (S)	$	reduce $S \rightarrow \epsilon$
5	$ (S) S	$	reduce S → (S) S
6	$ S	$	reduce $S' \rightarrow S$
7	$ S'	$	Accept

6

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Right Sentential Form (RSF) & Viable Prefixes

1. E' → E → E + n → n + n
2. S' → S → (S) S → (S) → ( )

- RSF splits the stack and the input during parsing.
- E, E + , E + n are all viable prefixes of RSF of E + n.

7

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Handle

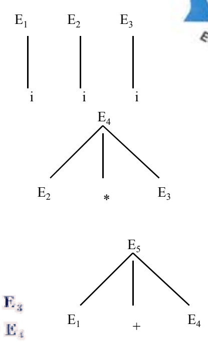
- Parser keeps shifting the terminals on the stack till the time it can perform reduction, that occurs when the RHS of a production rule matches. This string, position where it occurs and the production rule is called handle.
- Main task of a SR parser is to determine the next handle in the string.

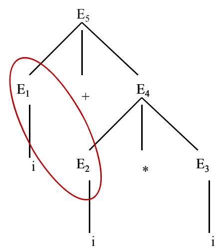
8

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Derivation Construction: $E \rightarrow E + E \mid E * E \mid i$
String = i + i * i

Stack	Input	Production
	i + i * i	shift
i	+ i * i	reduce $E_1 \rightarrow i$
E_1	+ i * i	shift +
E_1 +	i * i	shift i
E_1 + i	* i	reduce $E_2 \rightarrow i$
E_1 + E_2	* i	shift *
E_1 + E_2 *	i	shift i
E_1 + E_2 * i		reduce $E_3 \rightarrow i$
E_1 + E_2 * E_3		reduce $E_4 \rightarrow E_2 * E_3$
E_1 + E_4		reduce $E_5 \rightarrow E_1 + E_4$
E_5		





9

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Item

- An *item* is a grammar rule option with a distinguished position (indicated by a period or other symbol):

$$A \rightarrow \alpha \,.\, \beta$$

- The position in an item indicates that a parse has reached to the place in recognizing that rule α is then on the stack, and a β may be coming in the input (α is called a *viable prefix*).

- A stack state consists of the set of items which have "compatible" viable prefixes.

10

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Item Example 1

E' → E
E → E + n | n

E' → .E
E' → E.
E → .E + n
E → E. + n
E → E + .n
E → E + n.
E → .n
E → n.

Initial Items

Final Items

11

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Item Example 2

S' → S
S → (S) S | e

S' → .S
S' → S.
S → .(S) S
S → (.S) S
S → (S.) S
S → (S). S
S → (S) S.
S → .

12

Dr. D. M. Akbar Hussain
Department of Electronic Systems

FA of LR (0) Items

LR(0) items can be used as the states of FA which maintains the information about the parsing stack and the progress of shift-reduce parse.

Typically it starts with NFA and from this one can construct DFA using Subset construction or directly.

13

Dr. D. M. Akbar Hussain
Department of Electronic Systems

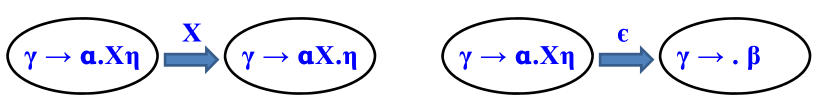
NFA Construction

Suppose $S \rightarrow \alpha.\gamma$ and imagine γ begins with a symbol X

$\gamma \rightarrow \alpha.X\eta$

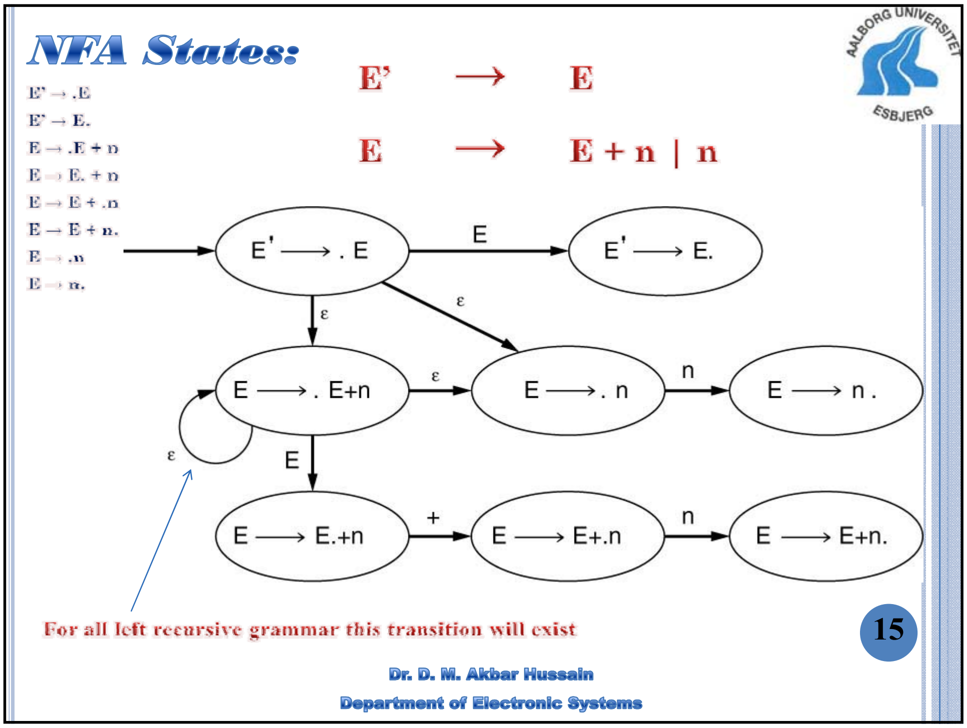
If X is terminal it is simple just make a transition to that terminal.

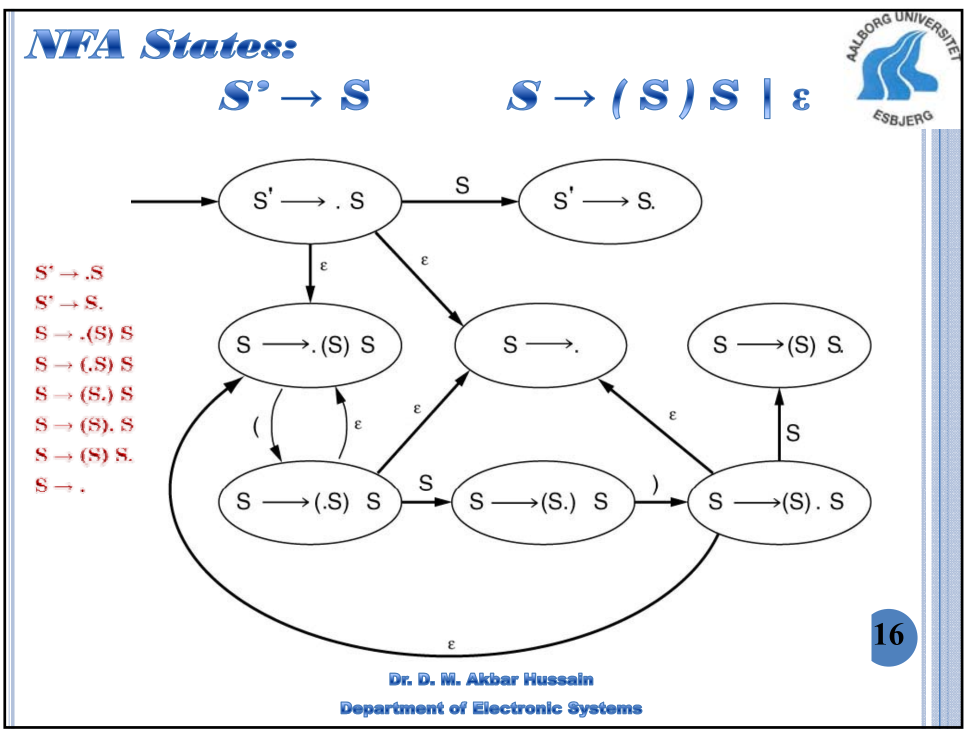
But if X is not a terminal, suppose $X \rightarrow \beta$, then a process starts by recognizing $.\beta$, so for each such case we need a transition of ϵ.



14

Dr. D. M. Akbar Hussain
Department of Electronic Systems


NFA States:
E' → .E
E' → E.
E → .E + n
E → E. + n
E → E + .n
E → E + n.
E → .n
E → n.
E' → E
E → E + n | n
E' ⟶ . E
E
E' ⟶ E.
ε
ε
E ⟶ . E+n
ε
E ⟶ . n
n
E ⟶ n .
ε
E
E ⟶ E.+n
+
E ⟶ E+.n
n
E ⟶ E+n.
For all left recursive grammar this transition will exist
15
Dr. D. M. Akbar Hussain
Department of Electronic Systems


NFA States:
S' → S
S → (S) S | ε
S' ⟶ . S
S
S' ⟶ S.
ε
ε
S' → .S
S' → S.
S → .(S) S
S → (.S) S
S → (S.) S
S → (S). S
S → (S) S.
S → .
S ⟶ . (S) S
S ⟶ .
S ⟶ (S) S.
(
ε
ε
ε
S
S ⟶ (.S) S
S
S ⟶ (S.) S
)
S ⟶ (S) . S
ε
16
Dr. D. M. Akbar Hussain
Department of Electronic Systems

Computing the DFA of sets of LR(0) items

- Add the augmentation item $S' \rightarrow .\ S$ to the start state of the DFA. Then add all the *initial items* for S to the state: $S \rightarrow .\ \alpha$. Continue by adding all the initial items for those non-terminals which appear right after the dot in any previous item (this is called the *closure* of the set of items).

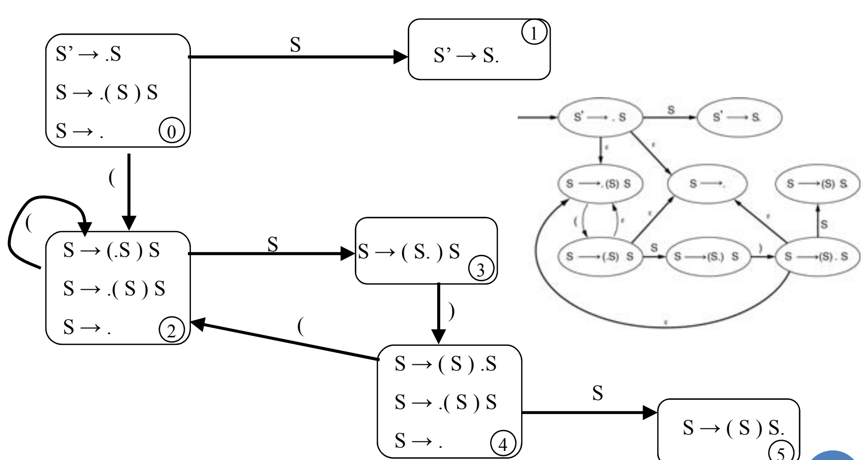
- Every symbol that comes immediately after the dot gives rise to a transition to a state generated by adding closure items to the item with the dot moved past that symbol.

17

Dr. D. M. Akbar Hussain
Department of Electronic Systems

DFA Set of States:

$S \rightarrow (\ S\ )\ S \mid \varepsilon$

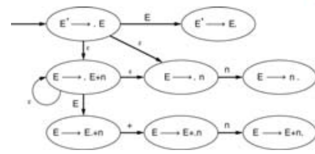


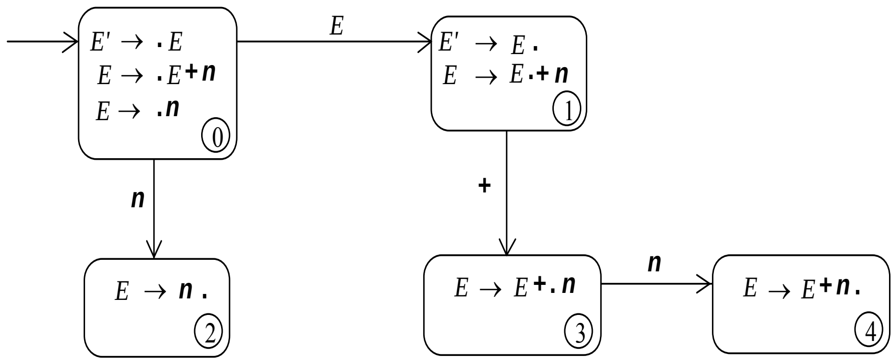
18

Dr. D. M. Akbar Hussain
Department of Electronic Systems

DFA Set of States:

$E \rightarrow E + n \mid n$





19

Dr. D. M. Akbar Hussain
Department of Electronic Systems

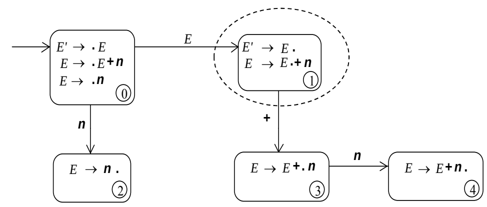
LR(0) Grammar

Every state is distinguishable as either it contains shift actions or reduce actions, any other possibility will make either **shift-reduce conflict** or **reduce-reduce conflict.** Which will indicate that it is not an LR(0) grammar, so these rules are exclusive.

20

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Sample Parse "n + n + n" string



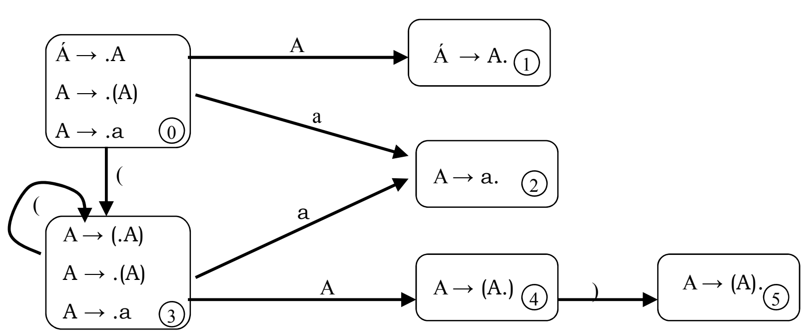
	Parsing stack	Input	Action
1	$ 0	*n* + *n* + *n* $	shift 2
2	$ 0 *n* 2	+ *n* + *n* $	reduce $E \to$ ***n***
3	$ 0 *E* 1	+ *n* + *n* $	shift 3
4	$ 0 *E* 1 + 3	*n* + *n* $	shift 4
5	$ 0 *E* 1 + 3 *n* 4	+ *n* $	reduce $E \to E$ + ***n***
6	$ 0 *E* 1	+ *n* $	shift 3
7	$ 0 *E* 1 + 3	*n* $	shift 4
8	$ 0 *E* 1 + 3 *n* 4	$	reduce $E \to E$ + ***n***
9	$ 0 *E* 1	$	accept

Dr. D. M. Akbar Hussain
Department of Electronic Systems

21

Example: A → (A) | a

The DFA of sets of items:

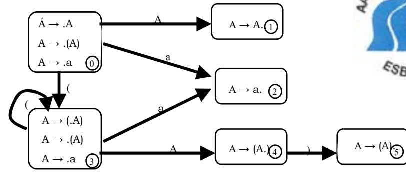


22

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Parsing Actions: ((a))

Grammar: A → (A) | a



	Parsing Stack	Input	Action
1	$0	((a))$	Shift
2	$0 (3	(a))$	Shift
3	$0 (3 (3	a))$	Shift
4	$0 (3 (3 a 2	))$	Reduce A → a
5	$0 (3 (3 A 4	))$	Shift
6	$0 (3 (3 A 4) 5	)$	Reduce A → (A)
7	$0 (3 A 4	)$	Shift
8	$0 (3 A 4) 5	$	Reduce A → (A)
9	$0 A 1	$	Accept

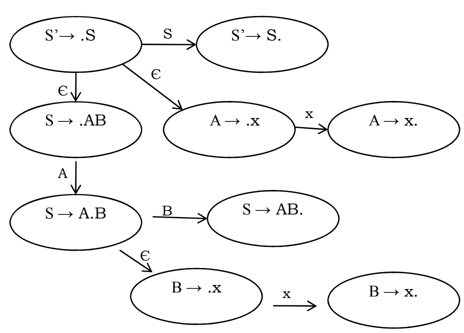
23

Dr. D. M. Akbar Hussain
Department of Electronic Systems

NFA LR(0) Items

Grammar: *S* → *A B*
A → x
B → x

S'→ .S
S → S.
S → .AB
S → A.B
S → AB.
A → .x
A → x.
B → .x
B → x.

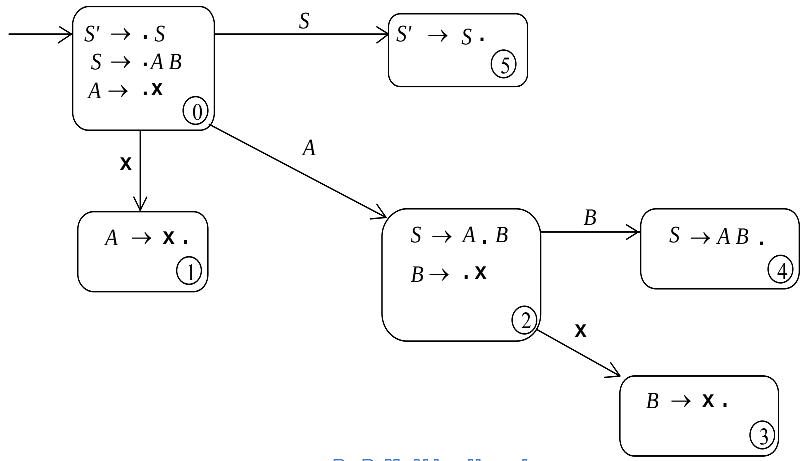


24

Dr. D. M. Akbar Hussain
Department of Electronic Systems

DFA States For:

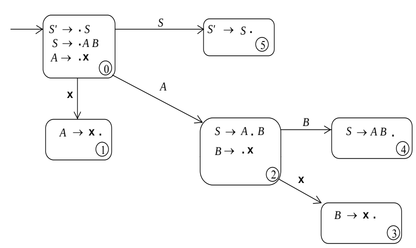
$S \rightarrow A\ B$
$A \rightarrow \mathbf{x}$
$B \rightarrow \mathbf{x}$



Dr. D. M. Akbar Hussain
Department of Electronic Systems

Sample Parse for "x x" string

$S \rightarrow A\ B$
$A \rightarrow \mathbf{x}$
$B \rightarrow \mathbf{x}$



	Parsing stack	Input	Action
1	$0	x x $	s1
2	$ 0 x 1	x $	r2 ($A \rightarrow$ x)
3	$ 0 *A* 2	x $	s3
4	$ 0 *A* 2 x 3	$	r3 ($B \rightarrow$ x)
5	$ 0 *A* 2 *B* 4	$	r1 ($S \rightarrow A\ B$)
6	$ 0 *S* 5	$	accept

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Yacc/Bison Parsing Tables

With the -v option ("verbose") Yacc generates a file y.output (Bison: <filename>.output) describing its parsing actions. For the same grammar

$$S \rightarrow A\ B$$
$$A \rightarrow \mathrm{x}$$
$$B \rightarrow \mathrm{x}$$

the output file looks as (next slide).

27

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Bison Table is Essentially the Same:

```
state 0
 'x'        shift, and go to state 1
 S go to state 5
 A go to state 2
state 1
 A  ->  'x' .(rule 2)
 $default reduce using rule 2 (A)
state 2
 S  ->  A . B (rule 1)
 'x'        shift, and go to state 3
 B go to state 4
```

```
state 3
 B  ->  'x'. (rule 3)
 $default            reduce using
    rule 3 (B)
state 4
 S  ->  A B . (rule 1)
 $default reduce using rule 1 (S)
state 5
   $ go to state 6
state 6
   $ go to state 7
state 7
   $default            accept
```

28

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Table Driven Bottom Up Parsing

- *ACTION | GOTO Table*

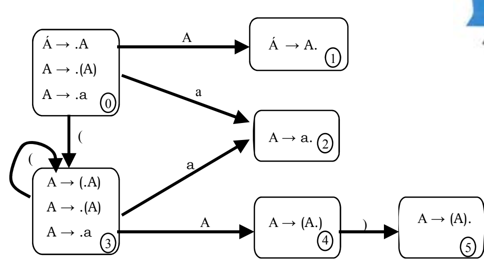
The set of DFA items & actions specified by the LR(0) algorithm can be combined into a parsing table, therefore it becomes a table driven parsing method.

29

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Table for Bottom Up Parsing

Grammar: A' → A
A → (A) | a

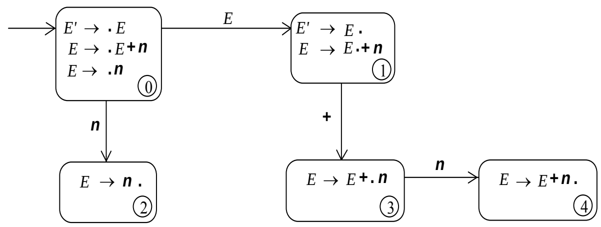


State	Action	Rule	Input			Goto
			(	a	)	A
0	Shift		3	2		1
1	Reduce	Reduce A' → A				
2	Reduce	Reduce A → a				
3	Shift		3	2		4
4	Shift				5	
5	Reduce	Reduce A → (A)				

30

Dr. D. M. Akbar Hussain
Department of Electronic Systems

LR(0) Parsing Table



State	Action	Rule	Input		Goto
			n	+	E
0	shift		2		1
1	shift/reduce	E '→ E		3	
2	reduce	E → n			
3	shift		4		
4	reduce	E → E + n			

Is this a LR(0) Grammar ?

31

Dr. D. M. Akbar Hussain
Department of Electronic Systems

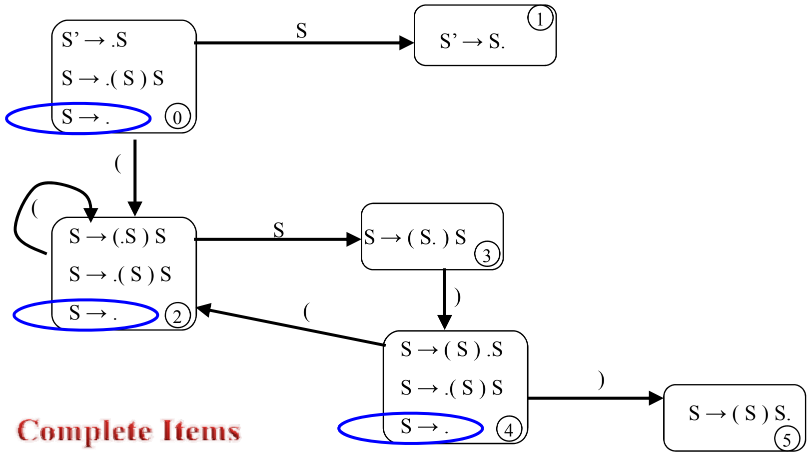
Decision Problems in BU Parsing

- **Shift-reduce conflicts:** Come from ambiguities and almost always the right disambiguating rule is to shift (dangling-else).
- **Reduce-reduce conflicts:** More difficult, bottom-up parsers try to resolve them using Follow set contexts.
- There are no **shift-shift conflicts.**

32

Dr. D. M. Akbar Hussain
Department of Electronic Systems

DFA Set of States: S → (S) S | ε



LR (0) Grammar ?

33

Dr. D. M. Akbar Hussain
Department of Electronic Systems

SLR (1) Parsing

The power is increased by two actions:

1. Consulting the input token before a shift is executed.
2. Follow set is used for reduction execution.

   For example if there is complete item A → x. and the next token in the input is in the Follow (A) use the A → x. rule for reduction.

34

Dr. D. M. Akbar Hussain
Department of Electronic Systems

SLR (1) Grammar

1. For any item A → x.Yc in a state s with a terminal Y, there is no complete item B → x. in s with Y in the Follow (B).
2. For any two complete items A → x., B → y. in s Follow (A) ∩ Follow (B) is empty.

35

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Reduce-Reduce Example

Grammar: $S \to A\ B$
$A \to \mathrm{x}$
$B \to \mathrm{x}$

Input: x x

Parse: (Follow(A) = {x}, Follow(B) = {$})

	Parsing stack	Input	Action
1	$	**x x** $	shift
2	$ **x**	**x** $	reduce $A \to$ **x**, reduce $B \to$ **x**
3	$ A	**x** $	shift
4	$ A **x**	$	reduce $B \to$ **x**
5	$ $A\ B$	$	reduce $S \to A\ B$
6	$ S	$	accept

36

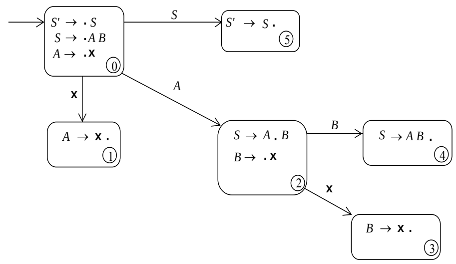
Dr. D. M. Akbar Hussain
Department of Electronic Systems

Parsing Table from the DFA

(1) $S \rightarrow A\ B$

(2) $A \rightarrow x$

(3) $B \rightarrow x$



State	Input		Goto		
	x	$	S	A	B
0	s1		5	2	
1	r2	r2			
2	s3				4
3	r3	r3			
4	r1	r1			
5		accept			

37

Dr. D. M. Akbar Hussain
Department of Electronic Systems

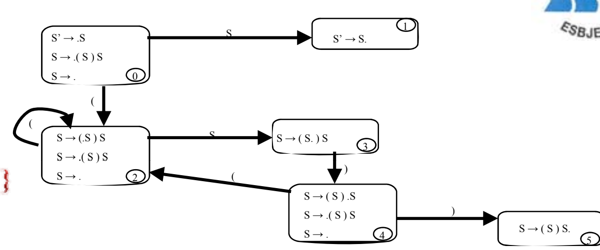
SLR (1) Parsing Table

S' → S

S → (S) S | ε

Follow (S') = { $ }

Follow (S) = { $,) }

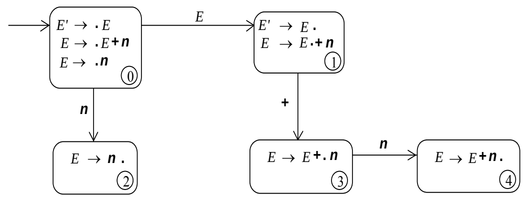


State	Input			Goto
	(	)	$	S
0	S2	R (S → ε)	R (S → ε)	1
1			Accept	
2	S2	R (S → ε)	R (S → ε)	3
3		S4		
4	s2	R (S → ε)	R (S → ε)	5
5		R (S → (S) S)	R (S → (S) S)	

38

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Parsing Table (*not LR(0)!*):



State	Input			Goto
	n	**+**	$	*E*
0	s2			1
1		s3	accept	
2		r ($E \rightarrow$ ***n***)	r ($E \rightarrow$ ***n***)	
3	s4			
4		r ($E \rightarrow E$ **+** ***n***)	r ($E \rightarrow E$ **+** ***n***)	

This table uses the SLR(1) rule: consult the Follow sets to decide between a shift and a reduce, or between two reduces:
Follow (E) = { + $ },
Follow (E') = {$}

39

Dr. D. M. Akbar Hussain
Department of Electronic Systems

DANGLING-ELSE:

Grammar:

(1) $S \rightarrow I$
(2) $S \rightarrow$ other
(3) $I \rightarrow$ if S
(4) $I \rightarrow$ if S else S

40

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Dangling-else Example:

Grammar: $S \rightarrow I \mid \text{o}$

$I \rightarrow \text{i}\, S \mid \text{i}\, S\, \text{e}\, S$

Input: i i o e o

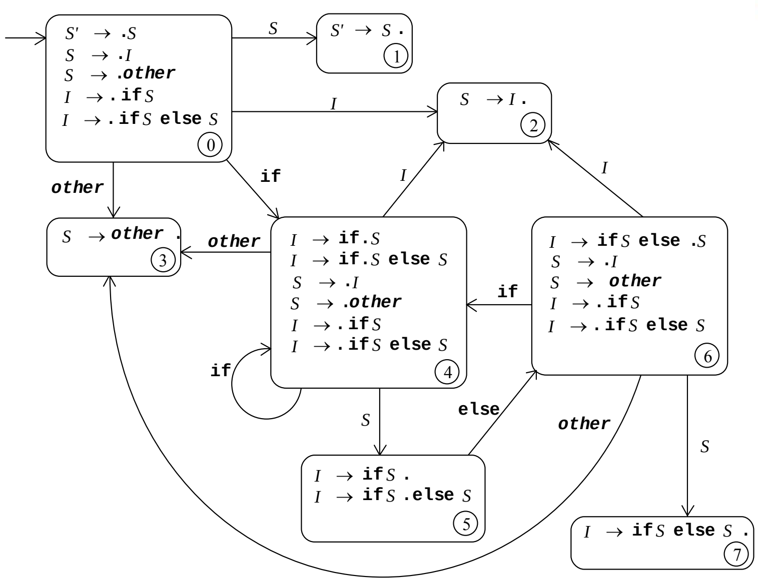
	Parsing stack	Input	Action
1	$	i i o e o $	shift
2	$ i	i o e o $	shift
3	$ i i	o e o $	shift
4	$ i i o	e o $	reduce $S \rightarrow \text{o}$
5	$ i i *S*	e o $	shift/reduce (shift)
6	$ i i *S* e	o $	shift
7	$ i i *S* e o	$	reduce $S \rightarrow \text{o}$
8	$ i i *S* e *S*	$	reduce $I \rightarrow \text{i}\, S\, \text{e}\, S$
9	$ i *I*	$	reduce $S \rightarrow I$
10	$ i *S*	$	reduce $I \rightarrow \text{i}\, S$
11	$ *I*	$	reduce $S \rightarrow I$
12	$ *S*	$	accept

41

Dr. D. M. Akbar Hussain

Department of Electronic Systems

DFA - Dangling-else:



42

Dr. D. M. Akbar Hussain

Department of Electronic Systems

First and Follow sets for Dangling-else

statement → *if-stmt* | *other*
if-stmt → if (*exp*) *statement else-part*
else-part → else *statement* | ε
exp → 0
First(*statement*) = {if, *other*}
First(*if-stmt*) = {if}
First(*else-part*) = {else, ε}
First(*exp*) = {0, 1}
Follow(*statement*) = {$,else}
Follow(*if-stmt*) = {$,else}
Follow(*else-part*) = {$,else}
Follow(*exp*) = {)}

43

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Parsing table (with conflict):

State	Input				Goto	
	if	else	other	$	S	I
0	s4		s3		1	2
1				accept		
2		r1		r1		
3		r2		r2		
4	s4		s3		5	2
5		s6/r3		r3		
6	s4		s3		7	2
7		r4		r4		

44

Dr. D. M. Akbar Hussain
Department of Electronic Systems

When Follow Set Fails

Stmt → call_stmt | assign_stmt
call_stmt → identifier
assign_stmt → Var := exp
Var → Var [exp] | identifier
exp → Var | num

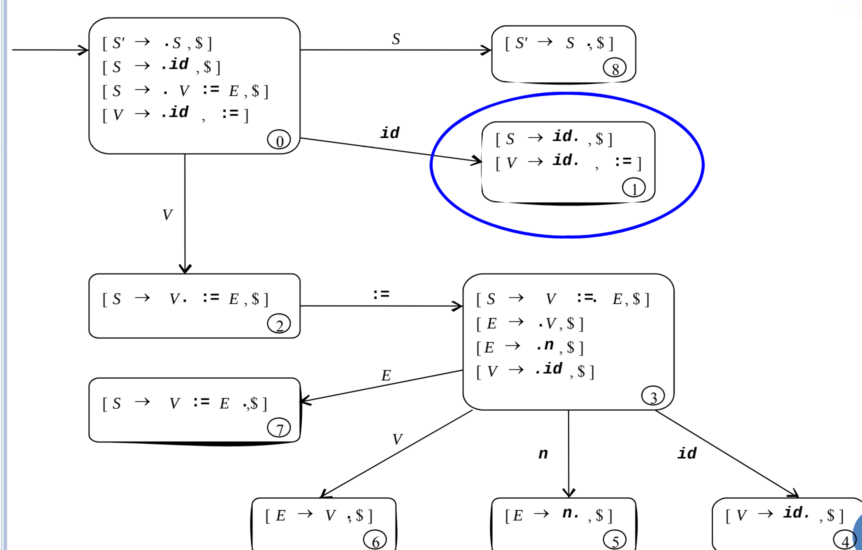
Simplified Version
S → id | V := E
V → id
E → V | n

Follow (S) = { $ }
Follow (V) = { $, := }

45

Dr. D. M. Akbar Hussain
Department of Electronic Systems

When Follow Set Fails



46

Dr. D. M. Akbar Hussain
Department of Electronic Systems

LR (1) & LALR (1) Parsing

LR (1) resolve problems of SLR(1) parser

Price : Complexity

Modification of LR (1) to LALR (1):

Retaining most functionality of LR (1).

Keeping the efficiency of SLR (1).

In SLR (1) look-ahead is applied after construction of DFA's of LR (0) items.

For LR (1) look-ahead are built into the DFA's.

47

Dr. D. M. Akbar Hussain
Department of Electronic Systems

LR (1) & LALR (1) Parsing

LR (1) item is: (LR (0) item + look-ahead token)

$[A \rightarrow \alpha.X\gamma, a]$

Given above LR (1) item if X is terminal then there will be transition to an item $[A \rightarrow \alpha X.\gamma, a]$.

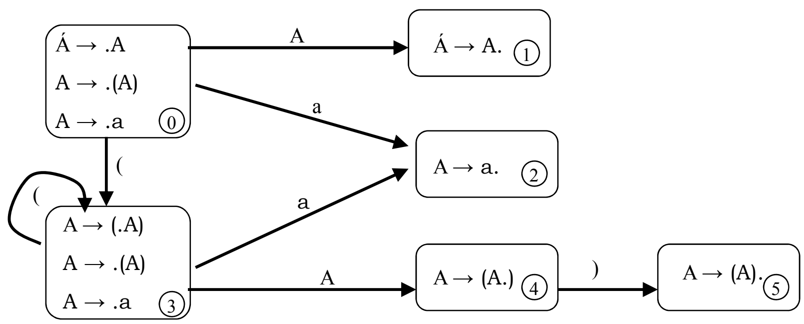
If X is non-terminal then, there will be ε transition to $[X \rightarrow .\beta, b]$ for every $X \rightarrow .\beta$ and for every b in First (γa).

48

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Example: A → (A) | a

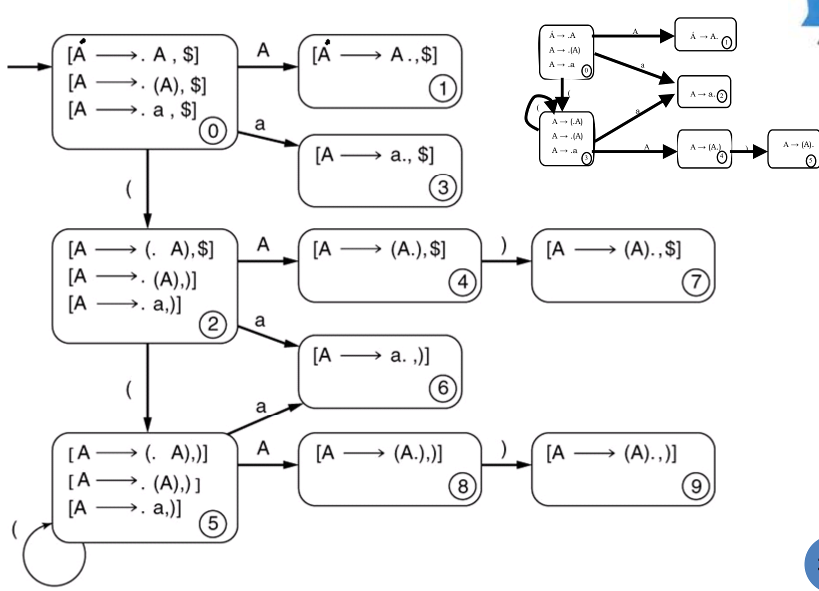
LR (0) Items



49

Dr. D. M. Akbar Hussain
Department of Electronic Systems

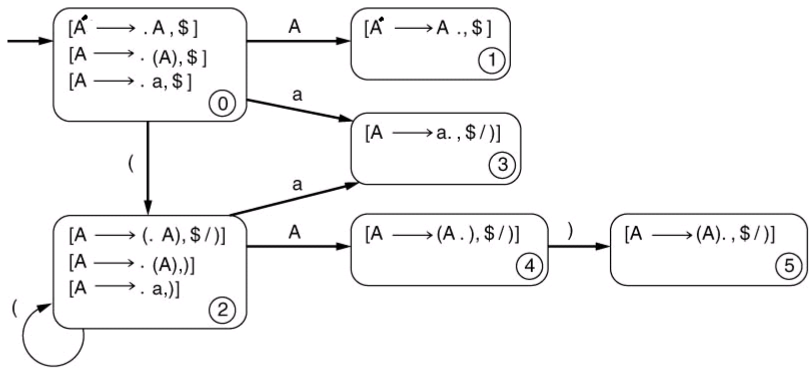
LR (1) DFA



50

Dr. D. M. Akbar Hussain
Department of Electronic Systems

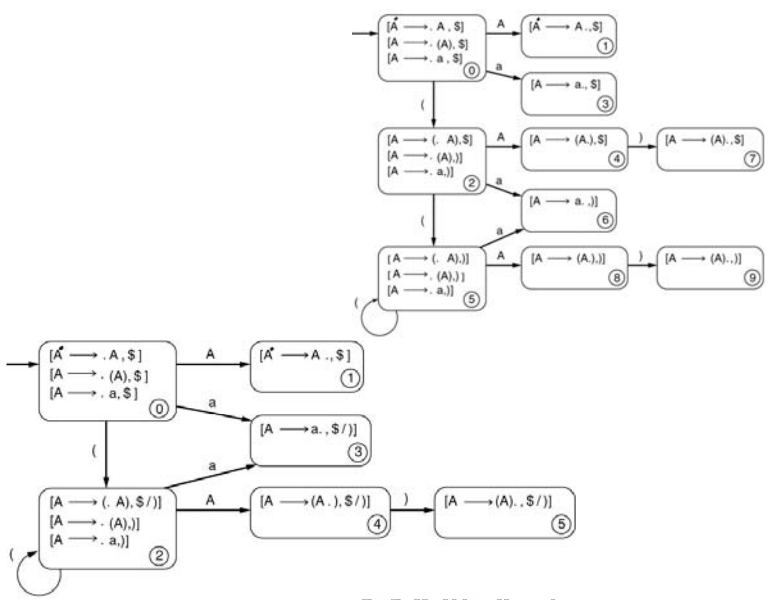
LALR (1) DFA



51

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Comparison between LR(1) & LALR(1) States



52

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Shift Reduce Parsers differ in the use of Follow Set Information

- **LR(0)** parsers never consult the look-ahead at all.
- **SLR(1)** parsers use the Follow sets as previously constructed.
- **LR(1)** parsers use context to split the Follow sets into subsets for different parsing paths (huge, inefficient parsers).
- **LALR(1)** parsers: like LR(1) but coarser subsets are used (achieves most of the benefit, but much smaller and faster).

53

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Stack Problem in BU Parsers

Grammar: S → (S) S | ε   Grammar: S → S (S) | ε

Input String: ( ) ( )

Parsing Stack (BU)	Input	Parsing Stack (TD)
$0	( ) ( ) $	$0
$0 (2	) ( ) $	$0 S 1
$0 (2 S 3	) ( ) $	$0 S 1 (2
$0 (2 S 3) 4	( ) $	$0 S 1 (2 S 3
$0 (2 S 3) 4 (2	) $	$0 S 1 (2 S 3) 4
$0 (2 S 3) 4 (2 S 3	) $	$0 S 1
$0 (2 S 3) 4 (2 S 3) 4	$	$0 S 1 (2
$0 (2 S 3) 4 (2 S 3) 4 S 5	$	$0 S 1 (2 S 3
$ 0 (2 S 3) 4 5	$	$0 S 1 (2 S 3) 4
$ 0 S 1	$	$0 S 1

54

Dr. D. M. Akbar Hussain
Department of Electronic Systems