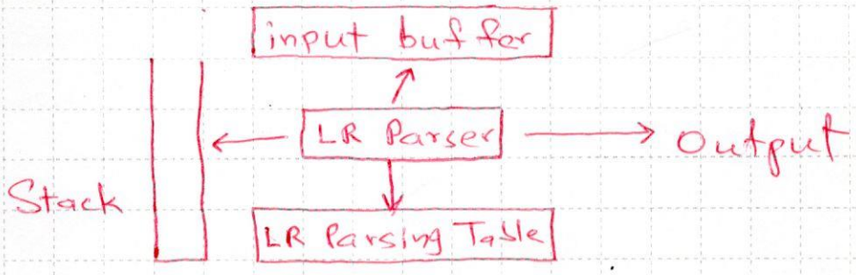
LR Parsers (LR(0))

LR(0), SLR(1) Simple LR, LALR(1) Look Ahead LR, CLR(1) Canonical LR

Least powerful ... Most powerful



Note: Construction of the parsing table varies for each type of the LR parsers.

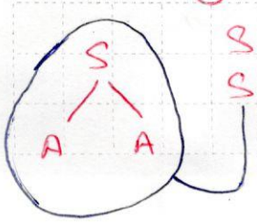
LR(0) and SLR(1) use Canonical collection of LR(0) items to construct the table.

For LALR(1) and CLR(1), canonical collection of LR(1) items is used.

$S \rightarrow AA$
$A \rightarrow aA \mid b$
Add $S' \rightarrow S$ to the start of the grammar to make it an augmented grammar.

Item is a production rule with a dot in it to indicate how much of the RHS has so far been recognized.

$S \rightarrow .AA$ Nothing has been recognized
$S \rightarrow A.A$
$S \rightarrow AA.$ Everything has been "



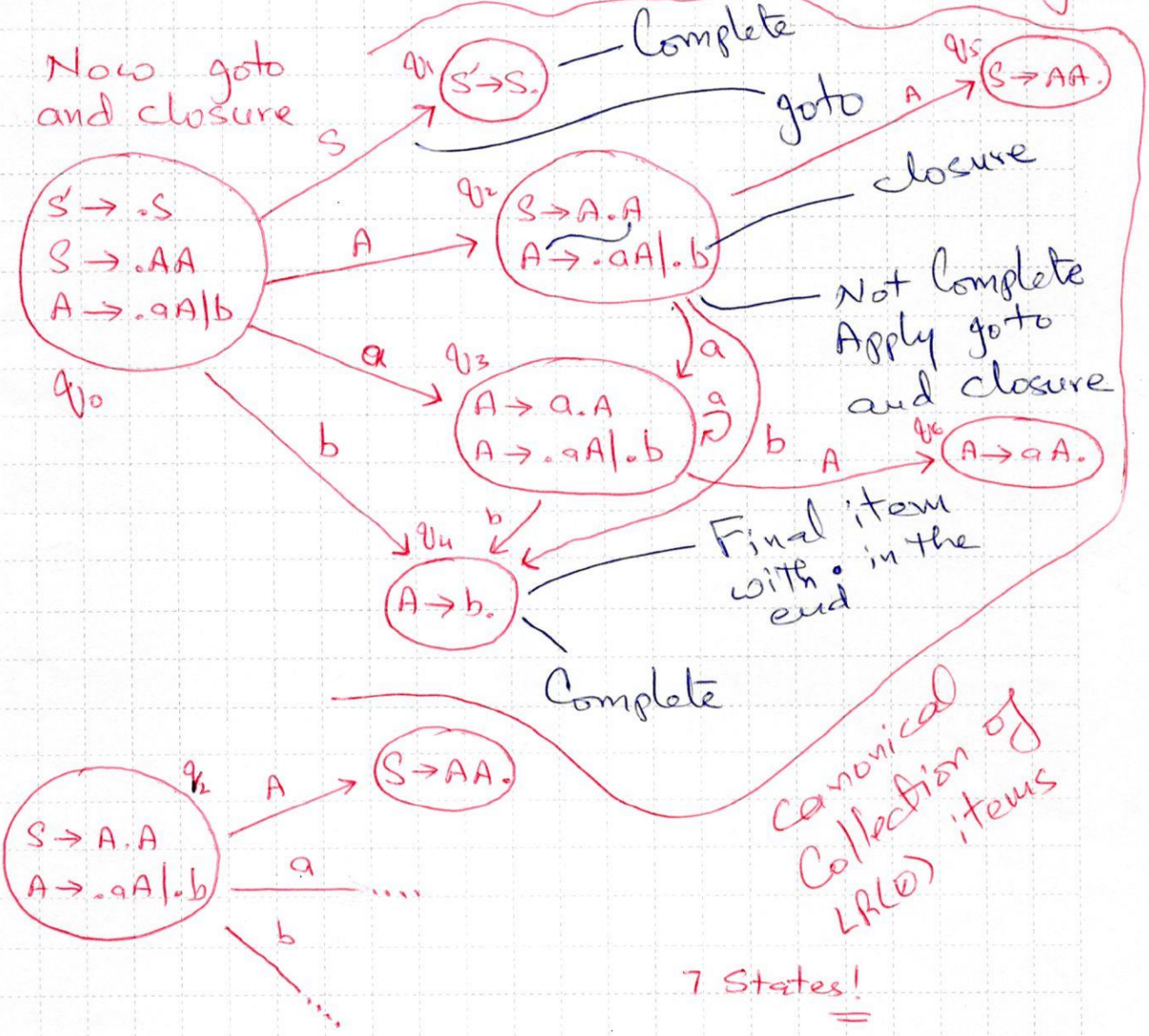
LR Parsers (LR(0))

$S' \to .S$ Production with . is an item

Apply closure (minimal extension)

$S' \to .S$
$S \to .AA$
$A \to .aA \mid .b$

Closure of $S' \to .S$
All NTs after . are taken care of



Canonical Collection of LR(0) items

7 States!

Final item: . is on/in the right/end

LR Parsers

Terminals / Variables

	Action			Goto	
	a	b	$	A	S
0	S_3	S_4		2	1
1					
2	S_3	S_4		5	
3	S_3	S_4		6	
4					
5					
6					

State

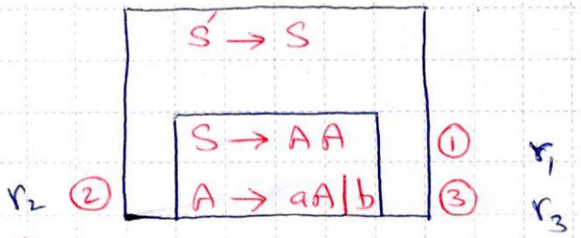
Shift

If a ~~state~~ transition goes to the final item, process changes for each type of the LR parsers.

	Action			Goto	
	a	b	$	A	S
0	S_3	S_4		2	1
1			Accept		
2	S_3	S_4		5	
3	S_3	S_4		6	
4	r_3	r_3	r_3		
5	r_1	r_1	r_1		
6	r_2	r_2	r_2		

Special production

Reduce



Input: aabb

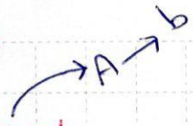
Append $: aabb$

Stack

0	a	3	a	3	b	4	

State ↙ ↘ Input

Now input is b and the table says r_3 meaning Reduce using production #3. Note that we are supposed to reduce previous b in this case. What is the length of the RHS of the production, Say x. Then pop 2x entries from stack.



Here, x = 1

0	a	3	a	3	A	

There is 3 below A in stack, So check state on A in the table. It is 6.

0	a	3	a	3	A	6

Note that the last entry, 6 is a state no.
Now, we are still at the last b of the input string.

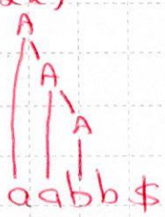
b on 6 is r_2. $A \to aA$ So, pop 4 cells (2x)

0	a	3	A	

Below A is 3 and 3,A is 6.

0	a	3	A	6

6, b is r_2, $A \to aA$, so pop 4 cells (2x)



0	A			

0, a is 2

0	A	2		

2, b is s_4, so shift b (input letter) and 4

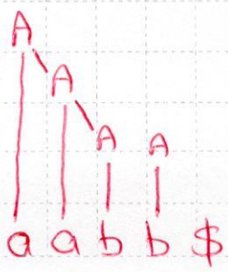
0	A	2	b	4

Next input is $

4, $ is r_3, $A \to b$, so pop 2 cells

0	A	2	A	

Below A is 2 and 2,A is 5



0	A	2	A	5

LR Parsers (LR(0)) 5 of 5

Next/current input is $

S, $ is r_1, S → AA, pop 4 symbols (2x)

0	S	

Below S is 0 and 0, S is 1

0	S	1	

Current input is $
and 1, $ is 'Accept'.

Blank cells represent ERROR.

Sometimes, LR(0) fails because we are not looking at the next letter in the input. We may have errors that will/may be detected later.