CSE 323: compiler Design

Bottom-up parsing:

↳ right-most derivation in reverse order

Ex.

Grammar

1. E → E+T
2. E → T
3. T → T*F
4. T → F
5. F → (E)
6. F → id

string: (id+id)*id

Derivation:

(id + id)*id

= (F+id)*id

= (T+id)*id

= (E+id)*id

= (E+F)*id

= (E+T)*id

= (E)*id

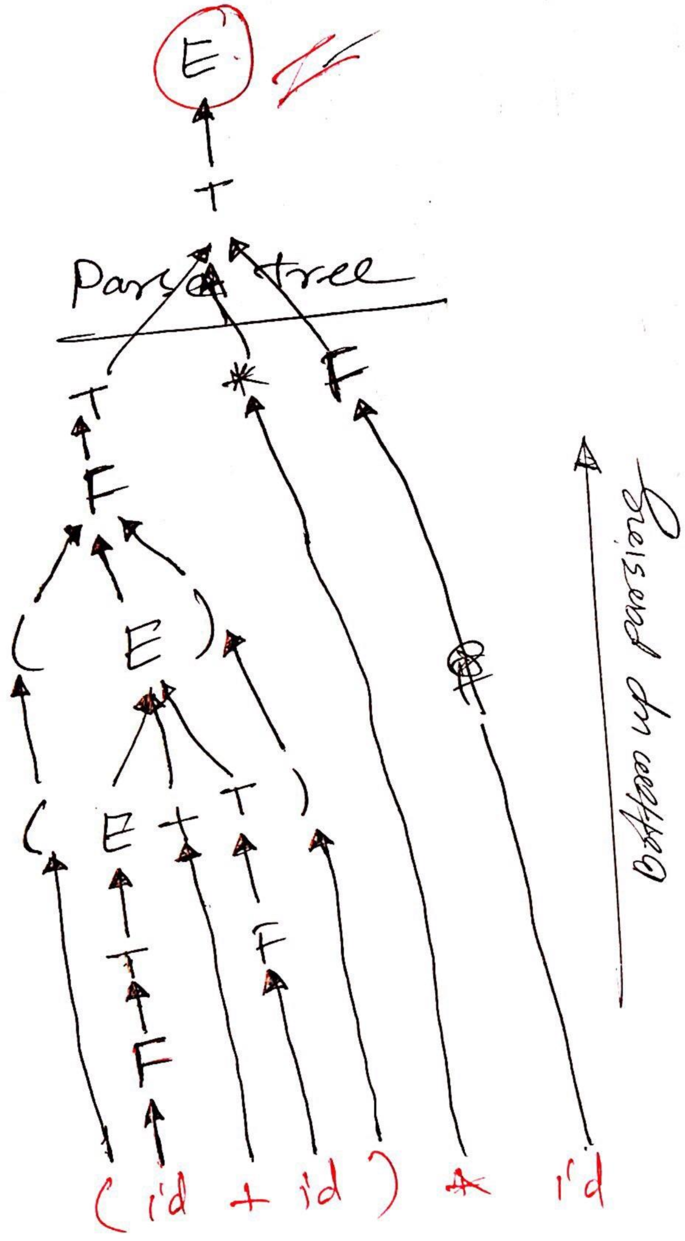
= F*id

= T*id

= T*F

= T

= E



CSE 323: Compiler Design

Shift-Reduce Parsing:

- → Bottom-up parsing
- → shift and reduce action

Operations:

1. shift
2. Reduction
3. Accept
4. Error

Ex:

Grammar:

1. $E \rightarrow E+T$
2. $E \rightarrow T$
3. $T \rightarrow T*F$
4. $T \rightarrow F$
5. $F \rightarrow (E)$
6. $F \rightarrow id$

Input string:

$id_1 + id_2$

Stack	Input	Action
\$	$id_1 + id_2$ \$	shift
\$ id_1	$+id_2$ \$	Reduce 6
\$ F	$+id_2$ \$	Reduce 4
\$ T	$+id_2$ \$	Reduce 2
\$ E	$+id_2$ \$	shift
\$ E+	id_2 \$	shift
\$ E+id_2	\$	Reduce 6
\$ E+F	\$	Reduce 4
\$ E+T	\$	Reduce 1
\$ E	\$	accept

H.W. Shift-Reduce Parsing

Q2:

$S \rightarrow aTRe$

$T \rightarrow Tbc \mid b$

$R \rightarrow d$

input string

abbcde

Conflicts during shift-Reduce parsing

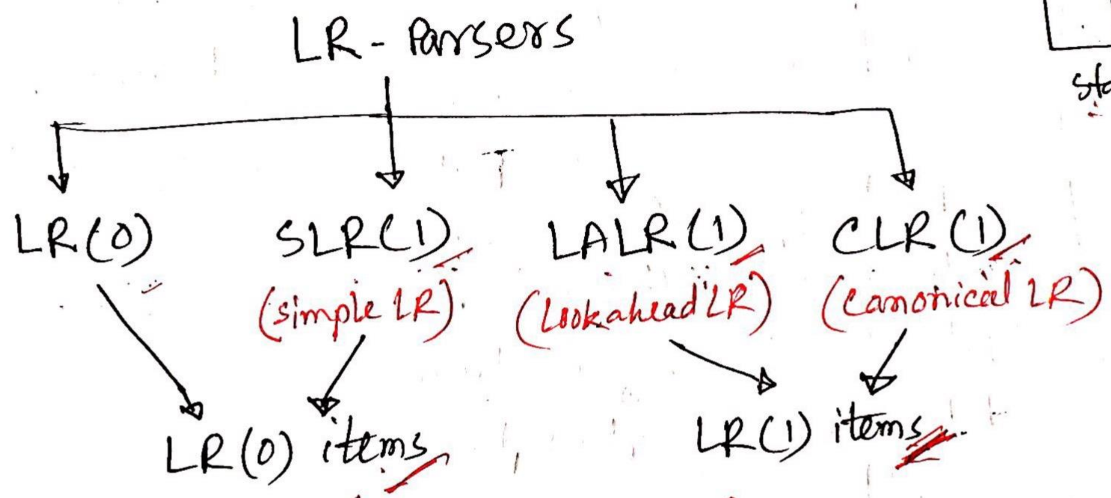
①. shift-Reduce conflicts

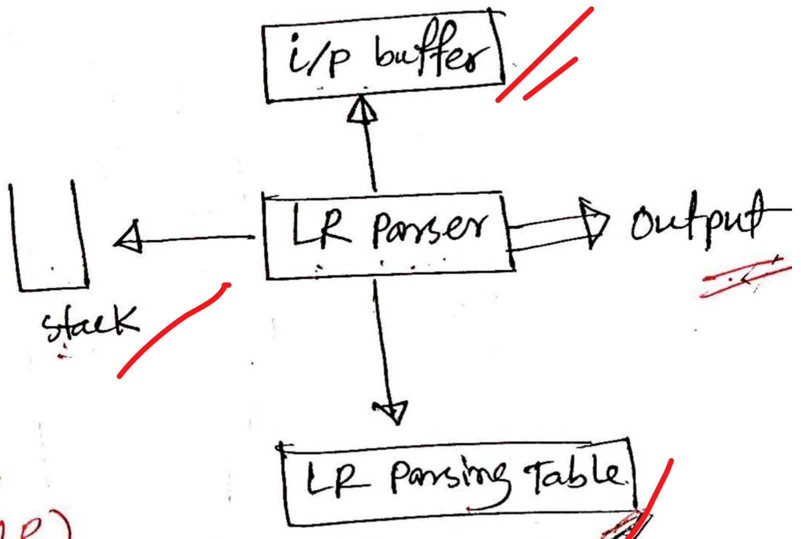
② Reduce-Reduce conflicts

CSE323: Compiler Design

Shift-Reduce Parsers main two categories:-

1. Operator-Precedence Parsers
   ↳ Simple, but only a small class of grammars
2. LR-Parsers
   ↳ covers wide range of grammars





CSE323: Compiler Design

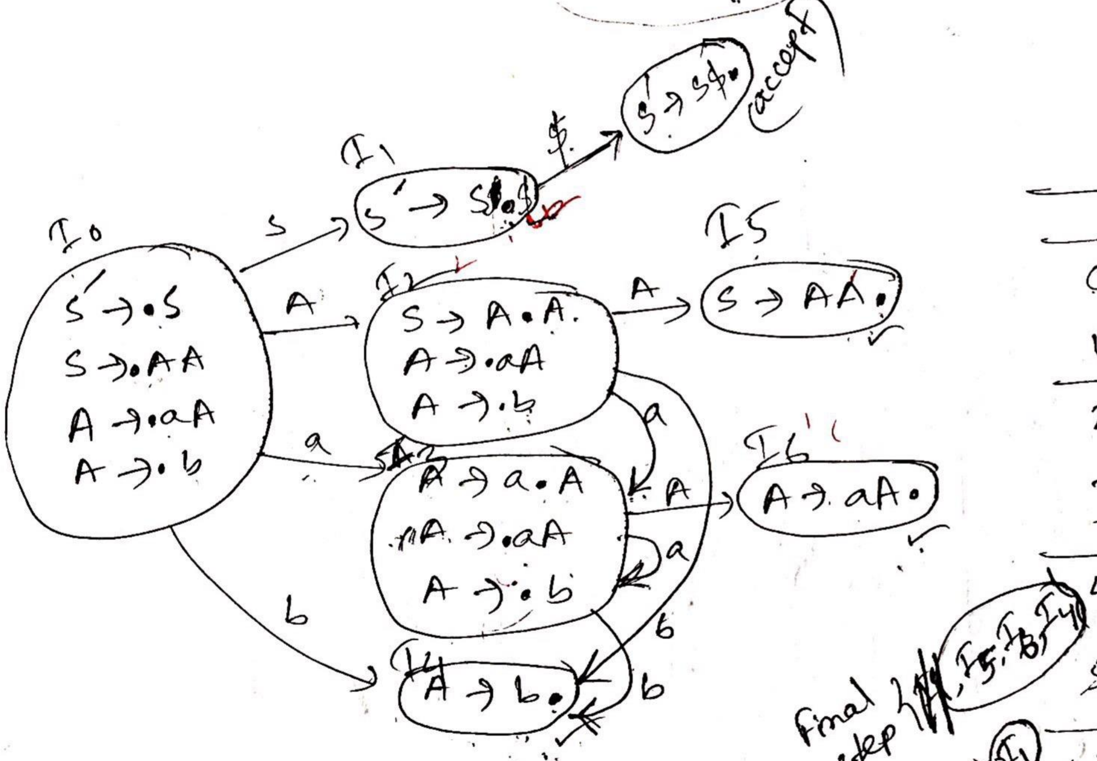
LR(0) Parsing:

G1: $\boxed{\begin{array}{l} S \to AA \\ A \to aA \mid b \end{array}}$

Augmented Grammar

0. $S' \to S$
1. $S \to AA$ //
2. $A \to aA$
3. $A \to b$ //

Input string: aabb$



Canonical Form / Tree

	action			goto	
	a	b	$	A	S
0	s3	s4		2	1
1			accept		
2	s3	s4		5	
3	s3	s4		6	
4	r3	r3	r3		
5	r1	r1	r1		
6	r2	r2	r3		

LR(0) Parsing Table

Input string

aabb

LR(0) parsing :

r_2 ; $\boxed{A} \rightarrow aA$

r_3 , $\boxed{A} \rightarrow b$. FOLLOW(A) = {a, b, $

r_1 , $\boxed{S} \rightarrow AA$ FOLLOW(S) = {$}

Stack	Input	Action
0	aabb$	S_3
0a3	abb$	S3
0a3a3	bb$	S4
0a3a3b4	b$	r3
0a3a3A6	b$	r2
0a3A6	b$	r2
0A2	b$	S4
0A2b4	$	r3
0A2A5	$	r1
0S1	$	(accept)

Successfully LR(0) parsing

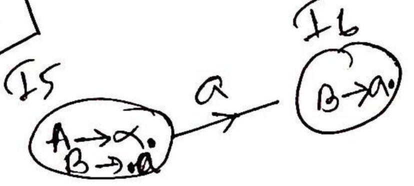
	action			goto	
	a	b	$	A	S
0	S3	S4		2	1
1			(accept)		
2	S3	S4		5	
3	S3	S4		6	
4	r3	r3	r3		
5			r1		
6	r2	r2	r2		

SLR(1) Parsing Table

CSE 323: Compiler Design

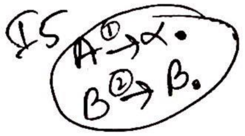
Conflicts in LR Parsing:

$\boxed{LR(0)}$



SR

	a	b
5	S6/r1	r1



RR

	a	b
5	r1/r2	r1/r2

$\boxed{SLR(1)}$

↳ conflicts during applying follow rules

Decision:

A grammar is not LR(0) if there is SR/RR conflicts

Same for SLR(1)

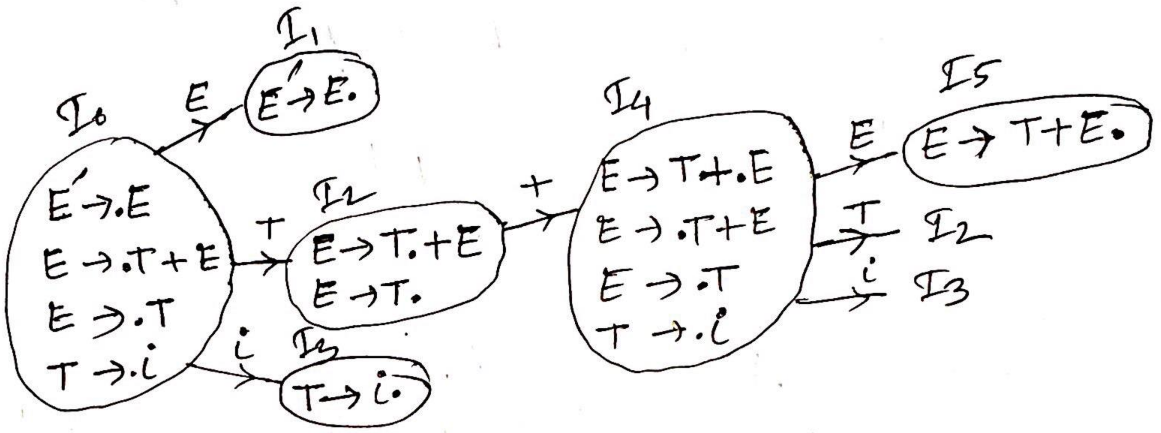
CSE 323 : Compiler Design

$\boxed{\begin{array}{l} LR(0)?\ \times \\ SLR(1)?\ \checkmark \end{array}}$

$\boxed{\begin{array}{l} E \to T+E \\ \quad\ /T \\ T \to i \end{array}}$

Augmented Grammar

0. $E' \to E$
1. $E \to T+E$
2. $E \to T$
3. $T \to i$



Follow.
$E' = \{\$\}$
$E = \{\$\}$
$T = \{+, \$\}$

	action			Goto	
	i	+	\$	E	T
0	s3			1	2
1			(accept)		
2	r2	s4/r2	r2		
3	r3	r3	r3		
4	s3			5	2
5	r1	r1	r1		

LR(0) Parsing Table

Here, S/R conflicts
its not a LR(0) grammar

	action			Goto	
	i	+	\$	E	T
0	s3			1	2
1			(accept)		
2		s4	r2		
3		r3	r3		
4	s3			5	2
5			r1		

SLR(1) Parsing Table
There's not SR/RR conflicts
So, its a SLR(1) grammar.