Find Push Down Automata that accepts the Context-free Grammar:

$P: \{ \quad S \to XY, \quad X \to aX \mid bX \mid a, \quad Y \to Ya \mid Yb \mid a \}$

$G = (\{S, X, Y\}, \{a, b\}, P, S)$

$[S \Rightarrow XY \Rightarrow aXY \Rightarrow aaY \Rightarrow aaYb \Rightarrow aaab]$

$\Gamma = V_N \cup \Sigma$

$\Rightarrow M = (\underset{Q}{\{q\}}, \underset{\Sigma}{\{a,b\}}, \underset{\Gamma}{\{S, X, Y, a, b\}}, \delta, q, S, \phi)$

Rule 1: $\delta(q, \epsilon, A) = \{(q, \alpha) \mid A \to \alpha\}$ (for each variable)

Rule 2: $\delta(q, a, a) = \{(q, \epsilon)\}$ (for each terminal)

$\Rightarrow \{1, 2, 3, \ldots\}$

R_1: $\delta(q, \epsilon, S) = \{(q, XY)\} \leftarrow$ as $S \to XY$

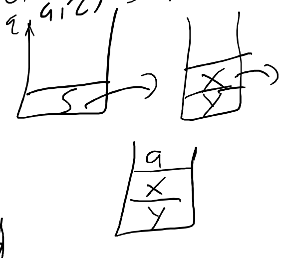
R_2: $\delta(q, \epsilon, X) = \{(q, aX), (q, bX), (q, a)\} \leftarrow$ as $X \to aX \mid bX \mid a$

R_3: $\delta(q, \epsilon, Y) = \{(q, Ya), (q, Yb), (q, a)\}$ as $Y \to Ya \mid Yb \mid a$

R_4: $\delta(q, a, a) = \{(q, \epsilon)\}$

R_5: $\delta(q, b, b) = \{(q, \epsilon)\}$

DPDA
1) ~~single element in RHS~~
2) if $\delta(q, a, z)$ is defined
if $\delta(q, \epsilon, z) = \phi$
$\delta(q, \epsilon, z)$ is defined
if $\delta(q, a, z) = \phi$



Consider $w = aaab$ for validation

$(q, aaab, S) \vdash (q, aaab, XY)$ $[\because (q, XY) \in \delta(q, \epsilon, S)]$

$\vdash (q, aaab, aXY)$ $[\because (q, aX) \in \delta(q, \epsilon, X)]$

$\vdash (q, aab, XY)$ $[\because (q, \epsilon) \in \delta(q, a, a)]$

$\vdash (q, aab, aY)$ $[\because (q, a) \in \delta(q, \epsilon, X)]$

$\vdash (q, ab, Y)$ $[\because (q, \epsilon) \in \delta(q, a, a)]$

$\vdash (q, ab, Yb)$ $[\because (q, Yb) \in \delta(q, \epsilon, Y)]$

$\vdash (q, ab, ab)$ $[\because (q, a) \in \delta(q, \epsilon, Y)]$

$\vdash (q, b, b)$ $[\because (q, \epsilon) \in \delta(q, a, a)]$

$\vdash (q, \epsilon, \epsilon)$ $[\because (q, \epsilon) \in \delta(q, b, b)]$

$0^4 1^4 2^4$ $n = 4$

$\underbrace{0\,0}_{u}\ \underbrace{0}_{v}\ \underbrace{0}_{w}\ \underbrace{1\,1\,1\,1}_{x}\ \underbrace{2\,2\,2\,2}_{y}$

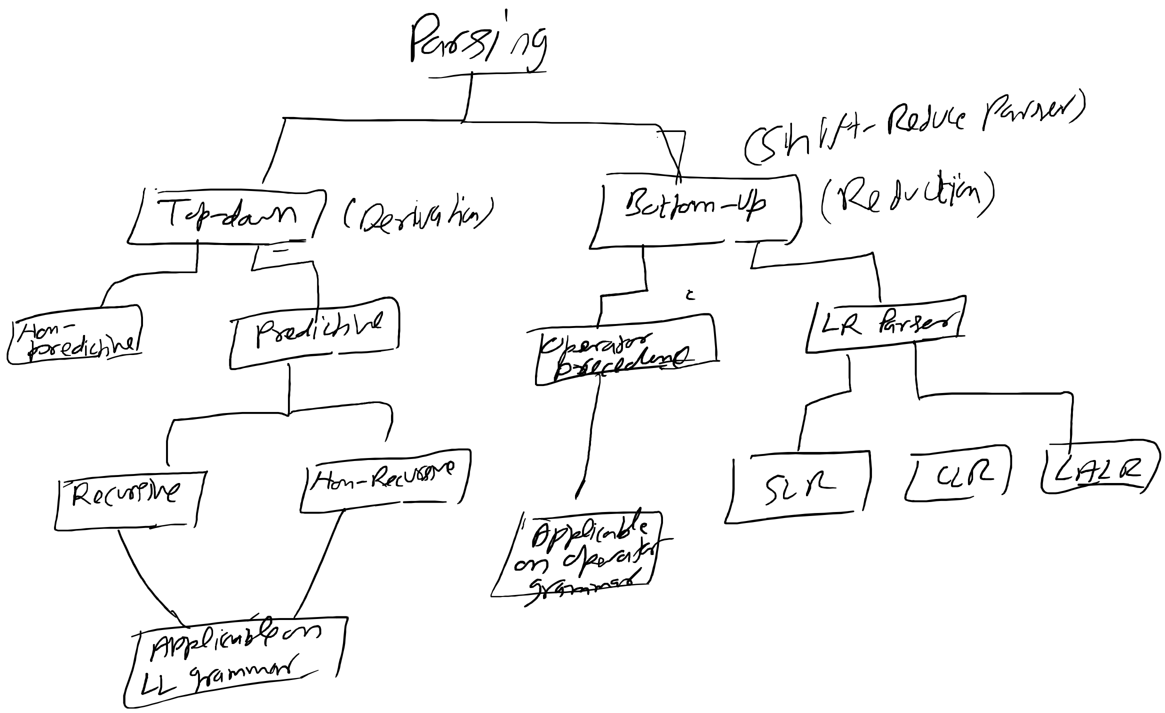
$n = 4$

$n = 4$

$\underbrace{}_{u}\ \underbrace{}_{v}\ \underbrace{}_{w}\ \underbrace{}_{x}\ \underbrace{}_{y}$

$|vwx| = 4$

Module-3 : Parsing



Non-predictive Topdown Parsing :

$S \to cAd$

$A \to ab \mid a$

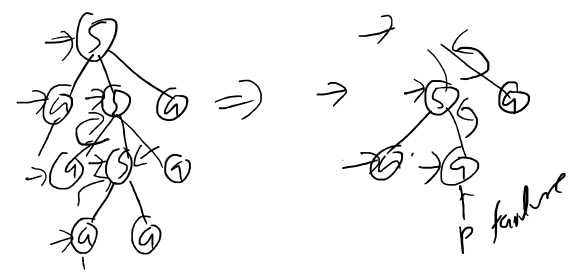
$w = cad$

cad

P.T. $S \to aSa \mid aa$ generates strings a^2, a^4 but not generate a^6 using non-predictive top down parsing

$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aaaaaa$

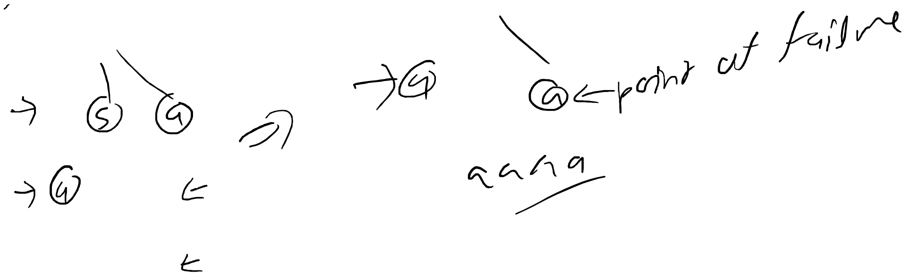
$S \Rightarrow aSa \Rightarrow aaSaa \Rightarrow aaaa\ldots$



(I) aaε

(II) aaaa

(III) aaaaaaε



point at failure

Predictive Top down Parsing:

→ ~~Grammar G must be unambiguous~~
→ " " " " non left recursive
→ " " " " left factoring] prerequisites

$E \to E+E \mid E*E \mid (E) \mid id$

$E \to E+E \mid E*E \mid (E) \mid id$

$\underbrace{id+id*id}$

$E \Rightarrow E+E \Rightarrow id+E \Rightarrow id+E*E \Rightarrow id+id*E \Rightarrow id+id*id$ (LMD1)

$E \Rightarrow E*E \Rightarrow E+E*E \Rightarrow id+E*E \Rightarrow id+id*E \Rightarrow id+id*id$ (LMD2)

$E \Rightarrow E+T \Rightarrow T+T \Rightarrow F+T \Rightarrow id+T \Rightarrow id+T*F \Rightarrow id+F*F \Rightarrow id+id*F \Rightarrow id+id*id$

$E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid id$

$id+id*id$

$\underbrace{id+id+id}$

$* > +$

$+ \doteq +$

$(2+3)+4 \quad (2\times 3)\times 4 = 6\times 4 = 24$

$+, -, *, /$ – left associative

$\uparrow, =$ – right associative

$2\uparrow\underbrace{3\uparrow 4} \Rightarrow 2^{81}$

$w = xyz \in L$

$\to xy^iz \in L \quad \forall i \geq 0$

$z \in L \quad |z| \geq n$ n is any positive constant

$z = uvwxy$

i) $|vx| \geq 1 \quad vx \neq \varepsilon$

ii) $|vwx| \leq n$

$uv^iwx^iy \in L \; \forall i \geq 0$

To prove L is not CFL, find i such that

$uv^iwx^iy \notin L$

$z = uvwxy$

Consider the grammar given below to construction predictive parser

$E \to E+E \mid E*E \mid (E) \mid id$ (Ambiguous)

$\Downarrow$

$E \to E+T \mid T$

$T \to T*F \mid F$ (Unambiguous)

$E \to E+T \mid T$
$T \to T*F \mid F$ (Unambiguous)
$F \to (E) \mid id$

$\Downarrow$ $\underline{A \to A\alpha / \beta} \Rightarrow \underline{A \to \beta A', \ A' \to \alpha A' \mid \epsilon}$

$E \to E+T \mid T$
$A = E, \ \alpha = +T, \ \beta = T$

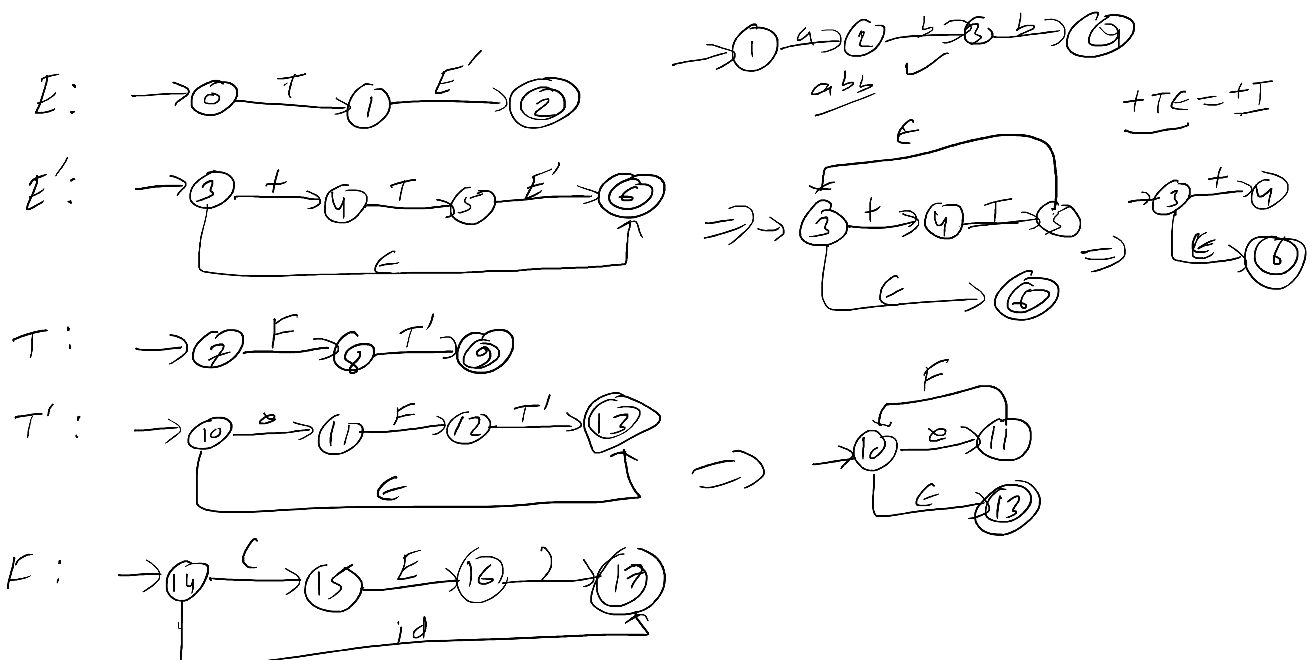
$E \to TE'$
$E' \to +TE' \mid \epsilon$
$T \to T\underline{*F} \mid F$
$A = T, \ \alpha = *F, \ \beta = F$
$T \to FT'$
$T' \to *FT' \mid \epsilon$
$F \to (E) \mid id$

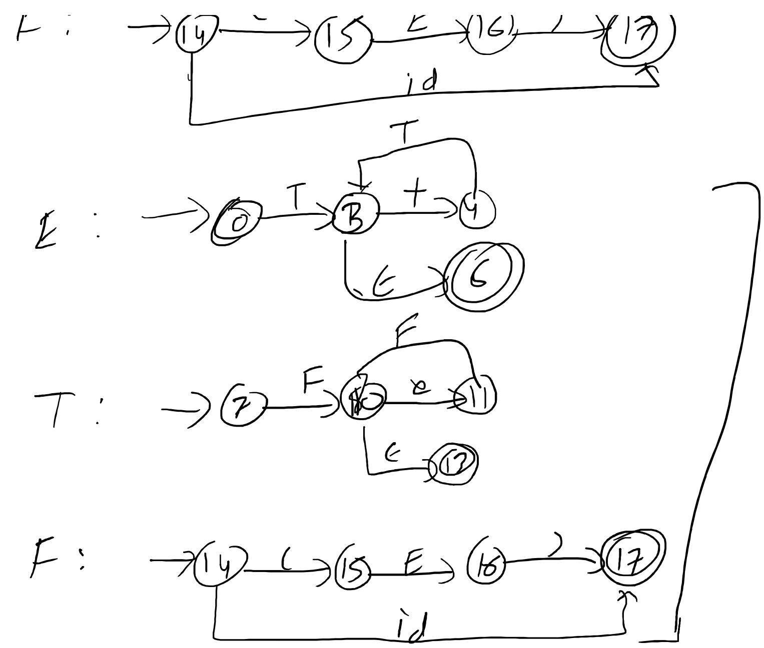
$\Rightarrow$

$E \to TE'$
$E' \to +TE' \mid \epsilon$
$T \to FT'$
$T' \to *FT' \mid \epsilon$
$F \to (E) \mid id$

Recursive Predictive Parsing

$E \to TE'$
$E' \to +TE' \mid \epsilon$
$T \to FT'$
$T' \to *FT' \mid \epsilon$
$F \to (E) \mid id$





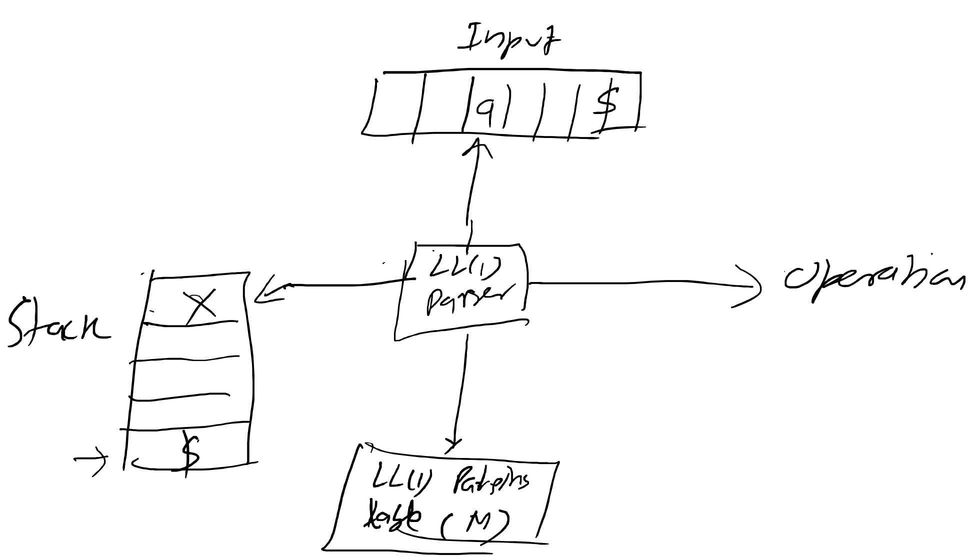
$w = id + id * id$

Non-recursive Predictive Parsing (LL(1) Parsing)

L L (1)

I/p Process Left to Right LMD

id + id * id



(1) $X = a = \$$ Success (i.e. string is syntactically valid)

... move i/p

(i) $X = a = \$$ Success (i.e. string is syntactically valid)

(ii) $X = a \neq \$$ Pop top element from the stack & move i/p pointer to next i/p element

(iii) $X \neq a$ Unsuccess i.e. given string is syntactically invalid string

(iv) if X is variable then refer entry in parsing table $M[X, a]$ $X \rightarrow uvw$, pop X from the stack & push wvu

(v) if X is variable & $M[X, a]$ is blank (error entry) string is invalid

Ex

$E \rightarrow E+E \mid E*E \mid (E) \mid id$ (Ambiguous)

$\Downarrow$

$E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$ (Unambiguous)
$F \rightarrow (E) \mid id$

$\Downarrow$

$A \rightarrow A\alpha \mid \beta \Rightarrow A \rightarrow \beta A'$
$A' \rightarrow \alpha A' \mid \epsilon$

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \epsilon$
$T \rightarrow FT'$ (non left-recursive)
$T' \rightarrow *FT' \mid \epsilon$
$F \rightarrow (E) \mid id$

$B \rightarrow \epsilon$

(Already left factored)

$X = \underbrace{aBC} = a\epsilon C = aC = a\epsilon = a$

FIRST(X):

(1) if X starts with a terminal a, then a included in FIRST(X)

(2) if $X \overset{*}{\underset{G}{\Rightarrow}} \epsilon$, then ϵ will be included in FIRST(X)

if $X = BC$, $B \overset{*}{\underset{G}{\Rightarrow}} \epsilon$, $C \overset{*}{\underset{G}{\Rightarrow}} \epsilon$

if

included in FIRST(X)

(3) if $X \to X_1 X_2 \cdots X_j X_{j+1} \cdots X_n$

$X_1 \overset{e}{\underset{m}{\Rightarrow}} \epsilon,\ X_2 \overset{e}{\underset{m}{\Rightarrow}} \epsilon \ \cdots\ X_j \overset{e}{\underset{m}{\Rightarrow}} \epsilon \qquad X_{j+1} \not\Rightarrow \epsilon$

$FIRST(X_{j+1} \cdots X_n)$ will be included in FIRST(X)

FOLLOW(B):

(1) if B is start symbol then \$ will be included in FOLLOW(B)

(2) if $A \to \alpha B \beta$, then $FIRST(\beta)$ will be included in FOLLOW(B) except ϵ

(3) if $A \to \alpha B \beta$, $\beta \overset{e}{\underset{m}{\Rightarrow}} \epsilon$ then FOLLOW(A) will be included in FOLLOW(B)

$E \to TE'$

$E' \to +TE' \mid \epsilon$

$T \to FT'$

$T' \to *FT' \mid \epsilon$

$F \to (E) \mid id$

$FIRST(E') \cap FOLLOW(E') = \phi$

$FIRST(T') \cap FOLLOW(T') = \phi$

$X = TE' \qquad FIRST(X) = \{ \} \ \phi$

$FIRST(E) = FIRST(TE') = \{ (, id \}$

$FIRST(E') = FIRST(+TE') \cup FIRST(\epsilon) = \{ +, \epsilon \}$

$FIRST(T) = FIRST(FT') = \{ (, id \}$

$FIRST(T') = FIRST(*FT') \cup FIRST(\epsilon) = \{ *, \epsilon \}$

$FIRST(F) = FIRST((E)) \cup FIRST(id) = \{ (, id \}$

$FOLLOW(E) = \{ \$,) \}$

$Follow(E) = \{\$,)\}$

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

$Follow(T) = \{+, \$,)\}$

$Follow(T') = \{+, \$,)\}$

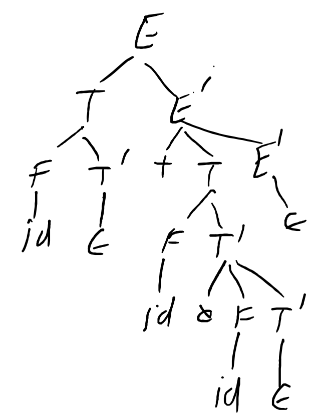
$Follow(F) = \{*, +, \$,)\}$

	+	*	(	)	id	$
E			$E \to TE'$		$E \to TE'$	
E'	$E' \to +TE'$			$E' \to \epsilon$		$E' \to \epsilon$
T			$T \to FT'$		$T \to FT'$	
T'	$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$		$T' \to \epsilon$
F			$F \to (E)$		$F \to id$	

LL(1)

Stack	Input	Production
$E	id+id*id$	
$E'T	id+id*id$	$E \to TE'$
$E'T'F	id+id*id$	$T \to FT'$
$E'T'id	id+id*id$	$F \to id$
$E'T'	+id*id$	$T' \to \epsilon$
$E'	+id*id$	$E' \to +TE'$
$E'T+	+id*id$	
$E'T	id*id$	$T \to FT'$
$E'T'F	id*id$	$F \to id$
$E'T'id	id*id$	
$E'T'	*id$	$T' \to *FT'$
$E'T'F*	*id$	
	id$	$F \to id$

Tree



id + id * id

$\$E'T'$	$*id\$$	$T' \to *FT'$
$\$E'T'F*$	$*id\$$	
$\$E'T'F$	$id\$$	$F \to id$
$\$E'T'id$	$id\$$	
$\$E'T'$	$\$$	$T' \to \epsilon$
$\$E'$	$\$$	$E' \to \epsilon$
$\$$	$\$$	

FIRST(tSA)

Consider the grammar:

$S \to iCtSA \mid a$

$A \to eS \mid \epsilon$

$C \to b$

Test whether it is LL(1) grammar and construct the predictive parsing table for it.

FIRST(S) = {i, a}
FIRST(A) = {e, ε}
FIRST(C) = {b}

Follow(S) = {$, e}
Follow(A) = {$, e}
Follow(C) = {t}

Prove that G is not LL(1)

$A \to eS \mid \epsilon$

FIRST(A) ∩ Follow(A) = {e, ε} ∩ {$, e} = {e} ≠ φ

So, Grammar is not LL(1)

	i	t	a	e	b	$
S	S→iCtSA		S→a			
A				A→eS A→ε		A→ε
C					C→b	

Bottom-Up Parser (Shift-Reduce Parser)

$E \to E + T \mid T$

$T \to T * F \mid F$

$F \to (E) \mid id$

$id + id * id$

$T + id * id \Rightarrow F + id * id \Rightarrow id + id * id$

RMD

$T + id * id \Rightarrow F + id * \ldots$

RMD

RMP γ_0 $id + id * id$ γ_1 $\Rightarrow F + id * id \Rightarrow T + id * id \Rightarrow E + id * id \Rightarrow E + F * id$ $\gamma_n = w$
$\Rightarrow E + T * id \Rightarrow E + T * F \Rightarrow E + T \Rightarrow E$

$\gamma_n = w$ γ_{n-1}

Handle prunning

~~$w = \gamma_n \Rightarrow \gamma_{n-1}$ form~~ β_n such that $A_n \to \beta_n$ (replace) — $F \to id$, 1st from left

Handle

$S \Rightarrow \gamma_1 \Rightarrow \gamma_2 \Rightarrow$ — — $\Rightarrow \gamma_n = w$ (derivation)

(reduction)

$\beta_n \to A_n$ to get γ_{n-1}

$A_{n-1} \to \beta_{n-1}$

$A_{n-2} \to \beta_{n-2}$ $\to \gamma_{n-3}$

and so on

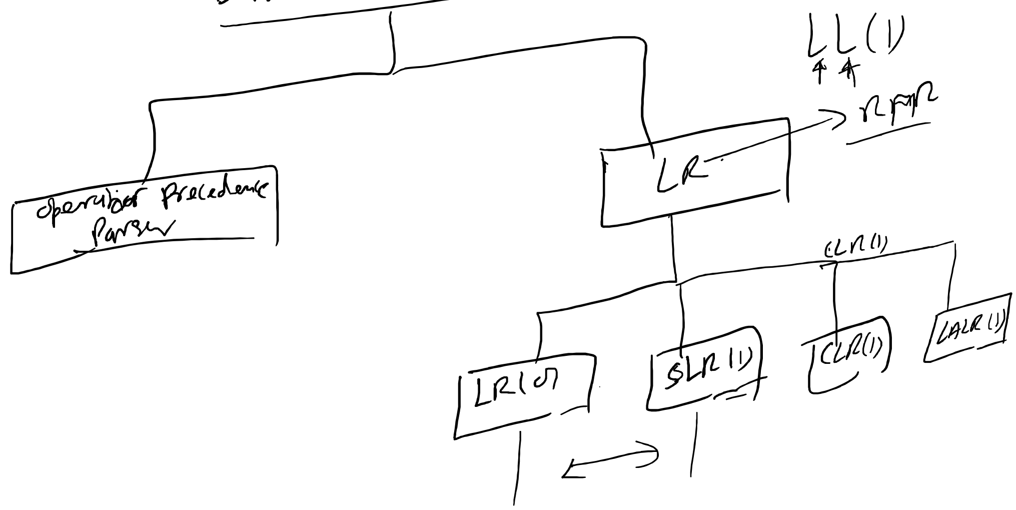
Stack	Input	operation
\$	id+id*id \$	Shift id
\$id	+id*id \$	Reduce by $F \to id$
\$F	+id*id \$	Reduce by $T \to F$
\$T	+id*id \$	Reduce by $E \to T$
\$E	+id*id \$	Shift +
\$E+	id*id \$	shift id
\$E+id	*id \$	Reduce by $F \to id$
\$E+F	*id \$	Reduce by $T \to F$
\$E+T	*id \$	shift *
\$E+T*	id\$	shift id
\$E+T*id	\$	[illegible] $F \to id$

\$ id+id*id \$

\$ E \$

$E+T	id$	shift
$E+T*	$	shift id
$E+T*id	$	Reduce $F \to id$
$E+T*F	$	Reduce $T \to T*F$
$E+T	$	Reduce $E \to E+T$
$E		

Bottom-up Parsing (Shift-Reduce Parser)



SLR(1) Parser (Simple LR)

Items $A \to XYZ$ (production)

$A \to \cdot XYZ$
$A \to X \cdot YZ$
$A \to XY \cdot Z$
$A \to XYZ \cdot$
(Items)

GoTo - Shifting dot (.) one position ahead

Closure - If after dot variable appears then explore all the productions of that variable and place dot at first position in all the productions.

Example

$E \to E+T \mid T$
$T \to T*F \mid F$

$F \to (E) \mid id$

Step I : Make the grammar augmented

augmented grammar

$E' \to E$ ← augmented production
$E \to E+T$
$E \to T$
$T \to T*F$
$T \to F$
$F \to (E)$
$F \to id$

Step II Compute cannonical collection of LR(0) Items

$I_0 = \{ E' \to \cdot E$
$E \to \cdot E+T$
$E \to \cdot T$ $\quad \{ \overset{2}{E}, \overset{2}{T}, \overset{1}{F}, \overset{1}{(}, \overset{1}{id} \}$
$T \to \cdot T*F$
$T \to \cdot F$
$F \to \cdot (E)$
$F \to \cdot id$
$\}$

$GOTO(I_0, E) = I_1 = \{ E' \to E\cdot$ $\quad \{ \overset{1}{+} \}$
$E \to E\cdot +T$
$\}$

$GOTO(I_0, T) = I_2 = \{ E \to T\cdot$ $\quad \{ \overset{1}{*} \}$
$T \to T\cdot *F$
$\}$

$FIRST(*F) \cap FOLLOW(E) = \phi$
$\{*\} \cap \{\$, +,)\} = \phi$

$GOTO(I_0, F) = I_3 = \{ T \to F\cdot \}$

$GOTO(I_0, () = I_4 = \{$
$F \to (\cdot E)$
$E \to \cdot E+T$
$E \to \cdot T$ $\quad \{ \overset{2}{E}, \overset{2}{T}, \overset{1}{F}, \overset{1}{(}, \overset{1}{id} \}$
$T \to \cdot T*F$
$T \to \cdot F$ $\quad \{ E \to T\cdot$
$F \to \cdot (E)$ $\quad T \to T\cdot *F \}$
$F \to \cdot id$
$\}$

$F \to (\cdot E)$

$F \to \cdot id$
$\}$

$F \to (\cdot E)$

$GoTo(I_0, id) = I_5 = \{ F \to id\cdot \}$

$GoTo(I_1, +) = I_6 = \{$
$E \to E + \cdot T$
$T \to \cdot T * F$
$T \to \cdot F$
$F \to \cdot (E)$
$F \to \cdot id$
$\}$

$\{T, F, (, id\}$

$GoTo(I_2, *) = I_7 = \{$
$T \to T * \cdot F$
$F \to \cdot (E)$
$F \to \cdot id$
$\}$

$\{F, (, id\}$

$GoTo(I_4, E) = I_8 = \{$
$F \to (E\cdot)$
$E \to E\cdot + T$
$\}$

$\{), +\}$

$GoTo(I_4, T) = I_2$

$GoTo(I_4, F) = I_3$

$GoTo(I_4, () = I_4$

$GoTo(I_4, id) = I_5$

$GoTo(I_6, T) = I_9 = \{$
$E \to E + T\cdot$
$T \to T\cdot * F$
$\}$

$\{*\}$

$FIRST(*F) \cap FOLLOW(E) = \phi$
$\{*\} \cap \{\$, +,)\} = \phi$

$GoTo(I_6, F) = I_3$

$GoTo(I_6, () = I_4$

$GoTo(I_6, id) = I_5$

$GoTo(I_7, F) = I_{10} = \{ T \to T * F\cdot \}$

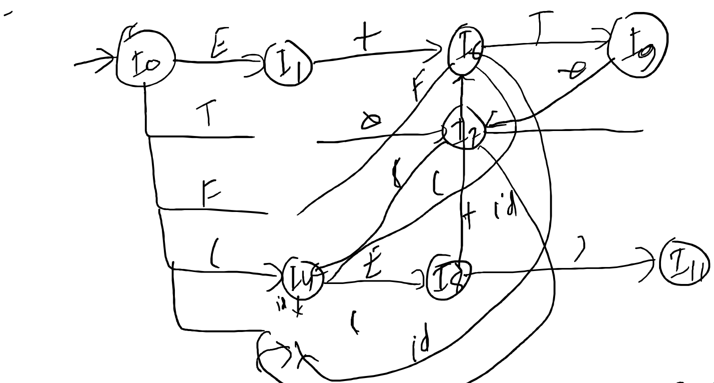
$GoTo(I_7, () = I_4$

$GoTo(I_7, id) = I_5$

$GoTo(I_8,)) = I_{11} = \{ F \to (E). \}$

$GoTo(I_8, +) = I_6$

$GoTo(I_9, *) = I_7$



SLR(1) Parsing table

	Action						GoTo		
States	+	*	(	)	id	$	E	T	F
I_0			s_4		s_5		1	2	3
I_1	s_6					accept			
I_2	r_2	s_7		r_2		r_2			
I_3	r_4	r_4		r_4		r_4			
I_4			s_4		s_5		8	2	3
I_5	r_6	r_6		r_6		r_6			
I_6			s_4		s_5			9	3
I_7			s_4		s_5				10
I_8	s_6				s_{11}				
I_9	r_1	s_7		r_1		r_1			
I_{10}	r_3	r_3		r_3		r_3			
I_{11}	r_5	r_5		r_5		r_5			

Shift
Reduce
accept

① $E \to E+T$ $\quad$ $Follow(E) = \{\$, +,)\}$

② $E \to T$ $\quad$ $Follow(T) = \{\$, +,), *\}$

③ $T \to T * F$

④ $T \to F$

⑤ $F \to (E)$

⑥ $F \to id$

Follow(F) = {\$, +,), *}

For LR(0) (LR- without referring follow) the reduce entries ($r_1, r_2, \ldots r_6$)

Parsing

Stack	Input
0	id+id*id \$
0 id 5	+ id*id \$
0 F 3	+ id*id \$
0 T 2	+ id*id \$
0 E 1	+ id*id \$
0 E 1 + 6	id*id \$
0 E 1 + 6 id 5	* id \$
0 E 1 + 6 F 3	* id \$
0 E 1 + 6 T 9	* id \$
0 E 1 + 6 T 9 * 7	id \$
0 E 1 + 6 T 9 * 7 id 5	\$
0 E 1 + 6 T 9 * 7 F 10	\$
0 E 1 + 6 T 9	\$
0 E 1	\$
	accept

[P.T. given grammar G is not SLR(1)

⇒ Prove that there exists Shift-Reduce conflict to prove grammar is not SLR(1)]

I_p : $A \to \alpha \cdot \beta$ (Shift)
$B \to \gamma \cdot$ (Reduce)

$FIRST(\beta) \cap FOLLOW(B) \neq \phi$
(Shift-Reduce conflict)

$E' \to E$
$E \to E+E \mid E*E \mid (E) \mid id$ (Ambiguous)

I_5 : $E \to E+E\cdot$

I_0

I_7: $E \to E*E\cdot$

(2+3)*4 = 20
2+3*4 = 14

FIRST(+E) ∪ FIRST(*E)
{+, *} ∩ {\$, +, *,)}
{+, *}

(2*3)+4
(2*3)*4

2*4 = 20 Follow(E) = {\$, +, *,)}

E $2+3$ ... $(2*3)*4$

$5\times4=20$ Follow$(E) = \{\$, +, *,)\}$

	+	*	(	)	id	\$
I_5	(S_7/r_1)	(S_{10}/r_1)				
I_7	(S_{10}/r_2)	(S_{11}/r_2)				

CLR(1) (Cannonical LR)

$E \to E+T \mid T$

$T \to T*F \mid F$

$F \to (E) \mid id$

Step I augmented grammar

$E' \to E$

$E \to E+T$

$E \to T$

$T \to F$

$F \to (E)$

$F \to id$

Step II Compute Cannonical collection of LR(1) items

core part

Lookahead part

I_0

$E' \to \cdot E, \$$

$E \to \cdot E+T, \$|+$

$E \to \cdot T, \$|+$

$T \to \cdot T*F, \$|+|*$

$T \to \cdot F, \$|+|*$

$F \to \cdot (E), \$|+|*$

$F \to \cdot id, \$|+|*$

First$(+T\$) = \{+\}$

$\{E, T, F, (, id\}$

$\{*\} \cup \{*\} = \{*\}$

$\{E, T, F, (, id\}$

I_1: $E' \to E\cdot, \$$

$E \to E\cdot+T, \$|+$

I_2: $E \to T\cdot, \$|+$

$T \to T\cdot*F, \$|+|*$

I_3 $T \to F\cdot, \$|+|*$

I_4 $F \to (\cdot E)$

Construct CLR(1)

$S \to CC$

$C \to cC \mid d$

Q. Design CLR(1) Parser for the following grammar

$S \rightarrow CC$

$C \rightarrow cC \mid d$

Step I make grammar augmented

augmented grammar

$S' \rightarrow S$ ← augmented production

$S \rightarrow CC$

$C \rightarrow cC$

$C \rightarrow d$

step II Compute cannonical collection of LR(1) items

$I_0 = S' \rightarrow \cdot S, \$$ FIRST(ε\$) = FIRST(\$) = {\$}

$S \rightarrow \cdot CC, \$$ FIRST(C\$) = {c, d}

$C \rightarrow \cdot cC, c|d$ {S, C, c, d}

$C \rightarrow \cdot d, c|d$

$I_1 = S' \rightarrow S\cdot, \$$

$I_2 = S \rightarrow C\cdot C, \$$ F(ε\$) = F(\$) = {\$}

$C \rightarrow \cdot cC, \$$ {C, c, d}

$C \rightarrow \cdot d, \$$

$I_3 = C \rightarrow c\cdot C, c|d$ {C, c, d}

$C \rightarrow \cdot cC, c|d$

$C \rightarrow \cdot d, c|d$

$I_4 = C \rightarrow d\cdot, c|d$

$I_5 = S \rightarrow CC\cdot, \$$

$I_6 = C \rightarrow c\cdot C, \$$ {C c d}

I_{36}

I_{47}

I_{89}

I_0

I_1

I_2

I_{36}

I_{47}

I_5

I_{89}

$C \rightarrow c\cdot C, c|d$

$I_6 = C \to c.C, \$$ $\quad \{C, c, d\}$

$C \to .cC, \$$

$C \to .d, \$$

$I_7 = C \to d., \$$

$I_8 = C \to cC., c/d$

$I_9 = C \to cC., \$$

States	Action			Go To	
	c	d	$	S	C
I_0	s_3	s_4		1	2
I_1			accept		
I_2	s_6	s_7			5
I_3	s_3	s_4			8
I_4	r_3	r_3			
I_5			r_1		
I_6	s_6	s_7			9
I_7			r_3		
I_8	r_2	r_2			
I_9			r_2		

$S \Rightarrow CC \Rightarrow cCC \Rightarrow cdC \Rightarrow cdd$

1. $S \to CC$
2. $C \to cC$
3. $C \to d$

Parsing

Stack	Input
0	cdd\$
0c3	dd\$
0c3d4	d\$
0c3C8	d\$
0C2	d\$
0C2d7	\$

0 C 2 C 0 d$

0 C 2 $

0 C 2 d 7 $

0 C 2 C 5 $

0 S 1 $

accept

LALR(1) (LookAhead LR) (LLR(1))

$S \to CC$

$C \to cC$

$C \to d$

step I make grammar augmented

$S' \to S$

$S \to CC$

$C \to cC$

$C \to d$

step II Compute canonical collection of LR(1) Items

I_0 $S' \to \cdot S, \$$ $\{S', C', c', d'\}$

$S \to \cdot CC, \$$

$C \to \cdot cC, c|d$

$C \to \cdot d, c|d$

I_1 $S' \to S\cdot, \$$

I_2 $S \to C\cdot C, \$$ $\{C', c', d'\}$

$C \to \cdot cC, \$$

$C \to \cdot d, \$$

I_3 $C \to c\cdot C, c|d$ $\{C', c', d'\}$

$C \to \cdot cC, c|d$

$C \to \cdot d, c|d$

I_4 $C \to d\cdot, c|d$

I_5 $S \to CC\cdot, \$$

I_6 $C \to c\cdot C, \$$ $\{C', c', d'\}$

$C \to \cdot cC, \$$

$C \to \cdot d, \$$

I_7 $C \to d\cdot, \$$

$C \to cC\cdot, c|d$

I_0

I_7 $C \to d\cdot, \$$

I_8 $C \to cC\cdot, c|d$

I_9 $C \to cC\cdot, \$$

I_{36} : $C \to c\cdot C, c|d|\$$
$C \to \cdot cC, c|d|\$$
$C \to \cdot d, c|d|\$$

I_{47} : $C \to d\cdot, c|d|\$$

I_{89} : $C \to cC\cdot, c|d|\$$

I_0
I_1
I_2
I_{36}
I_{47}
I_5
I_{89}
LALR states

① $S \to CC$
② $C \to cC$
③ $C \to d$

States	Action			GoTo	
	c	d	\$	S	C
I_0	S_{36}	S_{47}		1	2
I_1			accept		
I_2	S_{36}	S_{47}			5
I_{36}	S_{36}	S_{47}			89
I_{47}	r_3	r_3	r_3		
I_5			r_1		
I_{89}	r_2	r_2	r_2		

Parsing
Edd

I_3, I_6, I_9

I_{369}

P.T. given grammar is not LALR(1)

Soln : Prove that there exists Reduce-Reduce conflict (R-R)

I_5 : $A \to c\cdot, a$
$B \to d\cdot, c$

I_7 : $A \to c\cdot, d$
$B \to d\cdot, a$

I_{57} : $A \to c\cdot, a|d$
$B \to d\cdot, a|c$

	a	c	d	\$
I_{57}	r_5/r_6	r_6	r_5	

Reduce-Reduce conflict

I_{57} : $A \to c\cdot\ ,\ a|d$
$\underline{B \to d\cdot}\ ,\ \underline{a|c}$

$r_5\ ,\ r_6$

Reduce-Reduce Conflict

I_5 $A \to c\cdot\ ,\ a$
$B \to d\cdot\ ,\ b$

I_7 $A \to c\cdot\ ,\ d$
$B \to d\cdot\ ,\ c$

I_{57} $A \to c\cdot\ ,\ a|d$
$B \to d\cdot\ ,\ b|c$

	a	b	c	d	\$
I_{57}	r_5	r_6	r_6	r_5	

(No conflict)