

Parsing Theory

LHS = non-terminal on left side of production.

RHS = possibly empty string of terminals and/or non-terminals on right side of a production.

First-Set Theory

- Given application of a production $A \rightarrow \alpha$, which terminal symbols will reach the top of the parse stack when 0 or more subsequent productions are applied.
- We say that terminal symbols that reach the top of the parse stack are in the first set of the non-terminal A .

First Set Theory (cont)

For example, given the productions

$$A \rightarrow Bb$$

$$B \rightarrow cC \mid a$$

Where: non-terminals = $\{A, B, C\}$

terminals = $\{a, b, c\}$

start symbol = A

First Set $A = \{a, c\}$

By the way, First Set $B = \{a, c\}$ too.

First Set Theory (cont)

		c				
	B	C			B	a
A	b	b		A	b	b
\$	\$	\$		\$	\$	\$

Follow Set Theory

If a non-terminal goes to ε , what terminal symbols below it can rise up to the top of the stack.

First/Follow Set Theory

For example, given the productions

$$A \rightarrow Bb$$

$$B \rightarrow cC \mid a \mid \varepsilon$$

Where: non-terminals = $\{A, B, C\}$

terminals = $\{a, b, c\}$

start symbol = A

First Set $A = \{a, b, c\}$

First Set $B = \{a, c, \varepsilon\}$

b is in the follow set of B

First/Follow Set Theory (cont)

	B	
A	b	b
\$	\$	\$

Example 1

$$E \rightarrow TE'$$

$$E' \rightarrow +TE' \mid \varepsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow *FT' \mid \varepsilon$$

$$F \rightarrow (E) \mid \text{id}$$

Where :

non-terminals = $\{ E, E', T, T', F \}$

terminals = $\{ \text{id}, +, *, (,) \}$

goal symbol = E

Example 1 (cont)

	First Set	Follow Set
E	id, (\$,)
E'	+, ϵ	\$,)
T	id, (+, \$,)
T'	*, ϵ	+, \$,)
F	id, (*, +, \$,)

Follow Set Explanations

- \$ is always in the follow set of the goal symbol because \$ is under (follows) E at the start of parse.
-) is in the follow set of E because of $F \rightarrow (E)$.
- \$,) are in the follow set of E' because of $E \rightarrow TE'$

Follow Explanations (cont)

(T
E	E	E'
)))
\$	\$	\$

Table Construction

- Enter productions in table in columns of reachable terminals as well as first sets of reachable non-terminals beginning at the head of the RHS.
- If LHS non-terminal derives ε , enter that production in the table in columns defined by the follow set of the LHS non-terminal.

Parse Table

	id	+	*	()	\$
E	$E \rightarrow TE'$			$E \rightarrow TE'$		
E'		$E' \rightarrow +TE'$			$E' \rightarrow \varepsilon$	$E' \rightarrow \varepsilon$
T	$T \rightarrow FT'$			$T \rightarrow FT'$		
T'		$T' \rightarrow \varepsilon$	$T' \rightarrow *FT'$		$T' \rightarrow \varepsilon$	$T' \rightarrow \varepsilon$
F	$F \rightarrow \text{id}$			$F \rightarrow (E)$		

Derives ε

- A non-terminal derives ε if it can disappear from the top of the parse stack without placing anything on the parse stack.
- If a non-terminal derives ε , it is the follow set of that non-terminal that determines what is going to gravitate to the top of the parse stack.

Derives ε (cont)

- Add non-terminals that derive ε in one step to derives ε list.
- If the RHS of any production consists entirely of non-terminals, all of which derive ε , add that non-terminal to the derives ε list.

First Set

General Example finding first set of A:

$A \rightarrow BCDEFGHIJ$

- The first set of A contains the first set of all of the non-terminals from the beginning of the right hand side proceeding left to right up to and including the first terminal symbol or first set of a non-terminal that does not derive ϵ .
- Assume that F is the first symbol above (starting from the left of the RHS) that does not derive ϵ . Then the first set of A contains the first sets of B, C, D, E, and F.

First Set (cont)

$A \rightarrow BCDEFGHIJ$

If the entire RHS of the above production is non-terminals, all of which derive ε , then A derives ε and ε is in the first set of A .

The first sets of B , C , D , E , F , G , H , I , and J are also in the first set of A .

Follow Set

General Example finding follow set of NT D:

$A \rightarrow BCDEFGHIJ$

- Given any non-terminal on the RHS, the follow set of that non-terminal contains the first set (minus ϵ) of all of the consecutive non-terminals that follow the non-terminal proceeding left to right up to and including the first terminal symbol or first set of the first non-terminal that does not derive ϵ .
- Assume H is a non-terminal that does not derive ϵ , the follow set of D = $\{\text{first}(E) - \epsilon, \text{first}(F) - \epsilon, \text{first}(G) - \epsilon, \text{first}(H) - \epsilon\}$

Follow Set (cont)

General Example finding follow set of A

$A \rightarrow BCDEFGHIJ$

- The follow set of the LHS non-terminal is in the follow set of all consecutive non-terminals that derive ε starting at the extreme right end of the RHS and proceeding left until a terminal symbol or non-terminal that does not derive ε is reached.
- For example if H does not derive ε , the follow set of A is in the follow set of the non-terminals H, I, and J.

Table Construction

General Example:

$A \rightarrow BCDEFGHIJ$

- Place a production in columns specified by the first set of all non-terminals that derive ε (minus ε), starting at the left of the RHS and continuing up to and including a terminal symbol or the first set of a non-terminal that does not derive ε .
- If D is the first non-terminal from the left that does not derive ε , place the above production in the table in columns $(\text{first}(B) - \varepsilon, \text{first}(C) - \varepsilon, \text{and } \text{first}(D) - \varepsilon)$.

Table Construction (cont)

$A \rightarrow \varepsilon$

Place production $A \rightarrow \varepsilon$ in columns in follow set of A .

$A \rightarrow BCDEFGHIJ$, and all non-terminals on RHS derive ε

Place production $A \rightarrow BCDEFGHIJ$ in columns of follow set of A (note that the production should also be placed in columns of the first set of every non-terminal on the RHS – ε).

Example 2

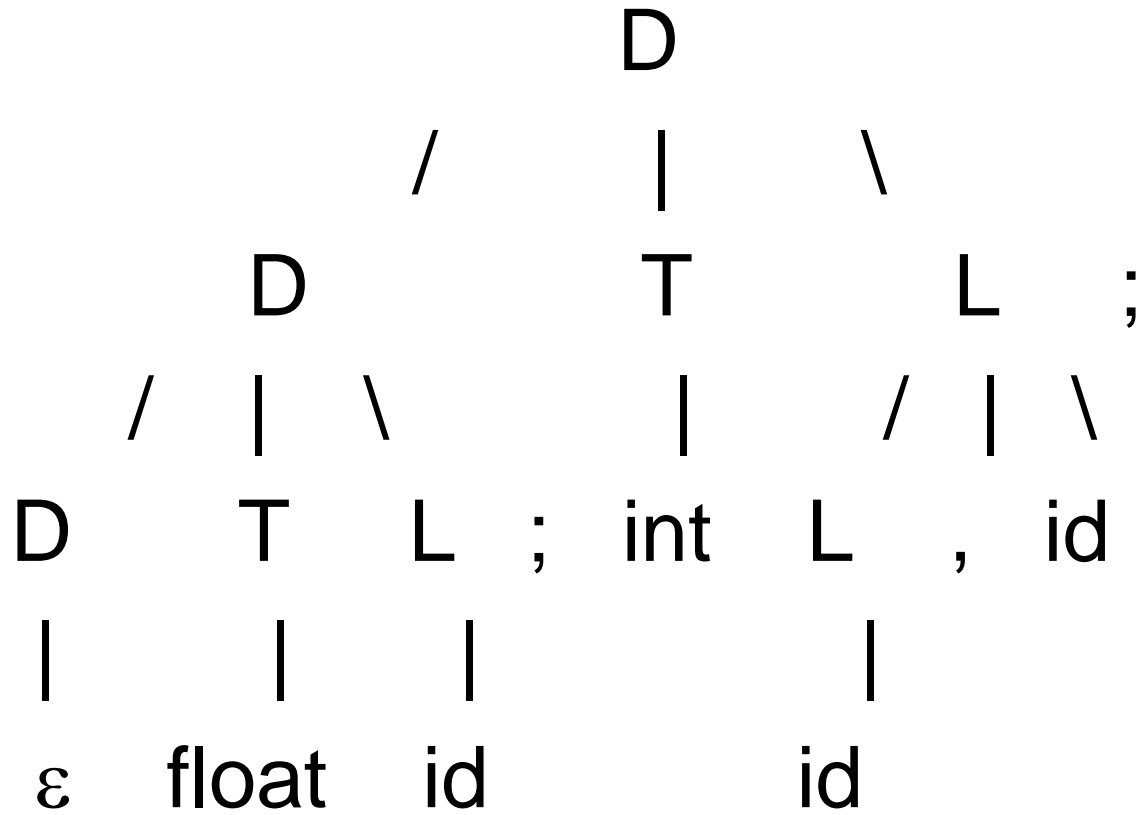
$D \rightarrow D T L ; | \varepsilon$

$T \rightarrow \text{int} | \text{float}$

$L \rightarrow L , \text{id} | \text{id}$

	First	Follow
D	int float ε	int float \$
T	int float	id
L	id	, ;

Parse Tree



Parse Table Example 2

	id	int	float	,	;	\$
D		$D \rightarrow D T L ;$ $D \rightarrow \epsilon$	$D \rightarrow D T L ;$ $D \rightarrow \epsilon$			$D \rightarrow \epsilon$
T		$T \rightarrow \text{int}$	$T \rightarrow \text{float}$			
L	$L \rightarrow L , \text{id}$ $L \rightarrow \text{id}$					

Elimination of Left Recursion (LR)

Replace productions of the form:

$$A \rightarrow A\alpha \mid \beta$$

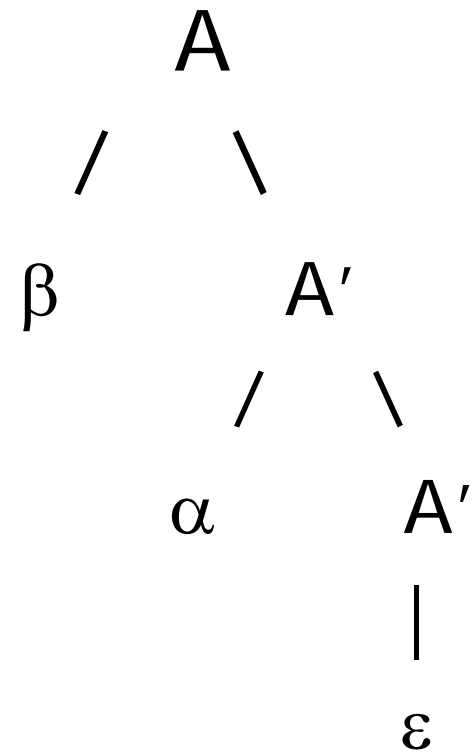
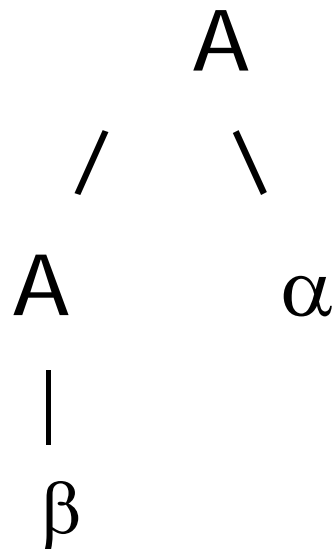
with:

$$A \rightarrow \beta A'$$

$$A' \rightarrow \alpha A' \mid \varepsilon$$

Elimination of LR (cont)

Parse of string: $\beta \alpha$



Example 1

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid \text{id}$$

In productions $E \rightarrow E + T \mid T$,
 $A = E$, $A' = E'$, $\alpha = +T$, $\beta = T$

In productions $T \rightarrow T * F \mid F$
 $A = T$, $A' = T'$, $\alpha = *F$, $\beta = F$

Example 1 (cont)

New productions:

$$E \rightarrow TE'$$

$$E' \rightarrow +TE' \mid \varepsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow *FT' \mid \varepsilon$$

$$F \rightarrow (E) \mid \text{id}$$

More on Elim of LR

What if LR occurs more than once?

Replace productions:

$$A \rightarrow A\alpha_1 \mid A\alpha_2 \mid \dots \mid A\alpha_n \mid \beta_1 \mid \beta_2 \mid \dots \mid \beta_m$$

With:

$$A \rightarrow \beta_1 A' \mid \beta_2 A' \mid \dots \mid \beta_m A'$$

$$A' \rightarrow \alpha_1 A' \mid \alpha_2 A' \mid \dots \mid \alpha_n A' \mid \varepsilon$$

Example 2

$$E \rightarrow E + T \mid E - T \mid T$$

$$T \rightarrow T * F \mid T / F \mid F$$

$$F \rightarrow (E) \mid \text{id}$$

In productions $E \rightarrow E + T \mid E - T \mid T$,

$$A = E, A' = E', \alpha_1 = +T, \alpha_2 = -T, \beta = T$$

In productions $T \rightarrow T * F \mid T / F \mid F$

$$A = T, A' = T', \alpha_1 = *F, \alpha_2 = /F, \beta = F$$

Example 2 (cont)

New productions:

$$E \rightarrow TE'$$

$$E' \rightarrow +TE' \mid -TE' \mid \varepsilon$$

$$T \rightarrow FT'$$

$$T' \rightarrow *FT' \mid /FT' \mid \varepsilon$$

$$F \rightarrow (E) \mid \text{id}$$

Dangling Else

Consider productions:

$$S \rightarrow i b t S \mid i b t S e S \mid s$$

Where $NT = \{S\}$, terminals = $\{i,b,t,e,s\}$,
Goal Symbol = S

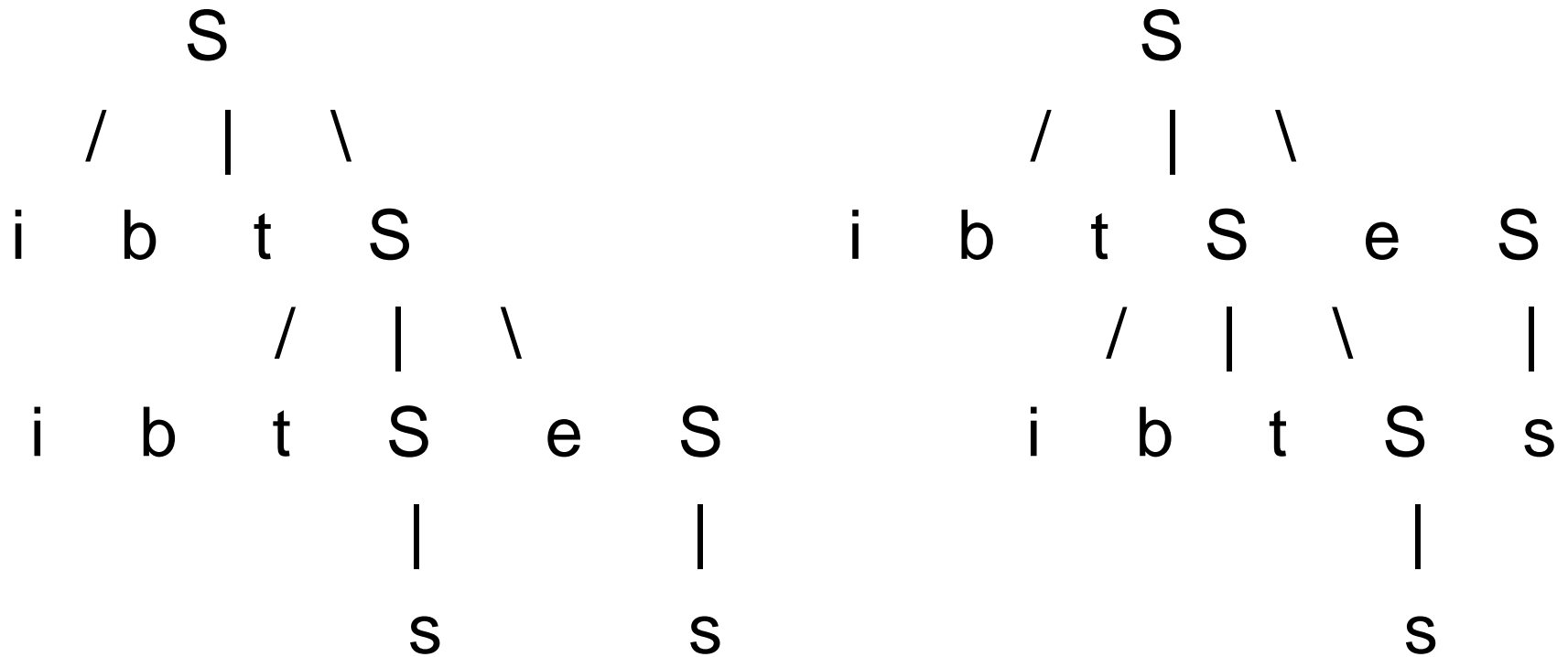
The above are productions for if-then
and if-then-else statements:

Parse Table

	i	b	t	s	e	\$
S	$S \rightarrow i b t S$ $S \rightarrow i b t S e S$			$S \rightarrow s$		

Dangling Else

Grammar is ambiguous because there are two parse trees for: i b t i b t s e s



Left Factoring

Replace productions of the form:

$$A \rightarrow \alpha \beta \mid \alpha \gamma$$

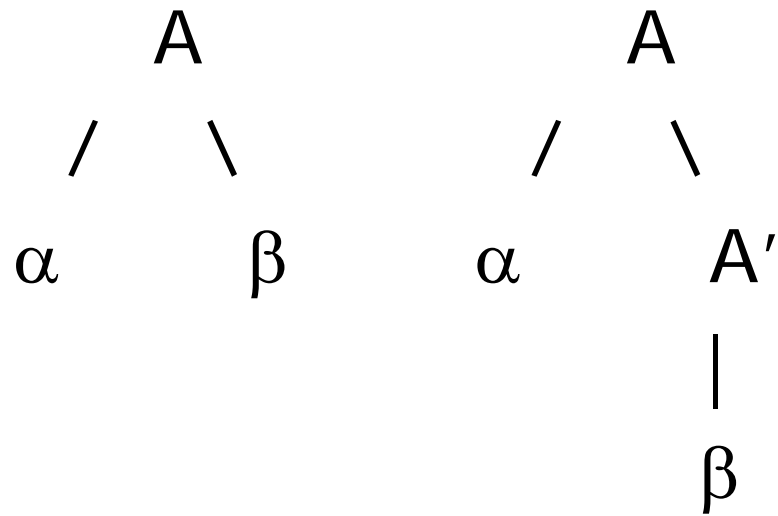
with:

$$A \rightarrow \alpha A'$$

$$A' \rightarrow \beta \mid \gamma$$

Left Factoring (cont)

Parse of string: $\alpha \beta$



Left Factoring (cont)

For:

$S \rightarrow i b t S \mid i b t S e S \mid s$

let:

$A = S, A' = S', \alpha = i b t S, \beta = \epsilon, \gamma = e S$

Left Factoring (cont)

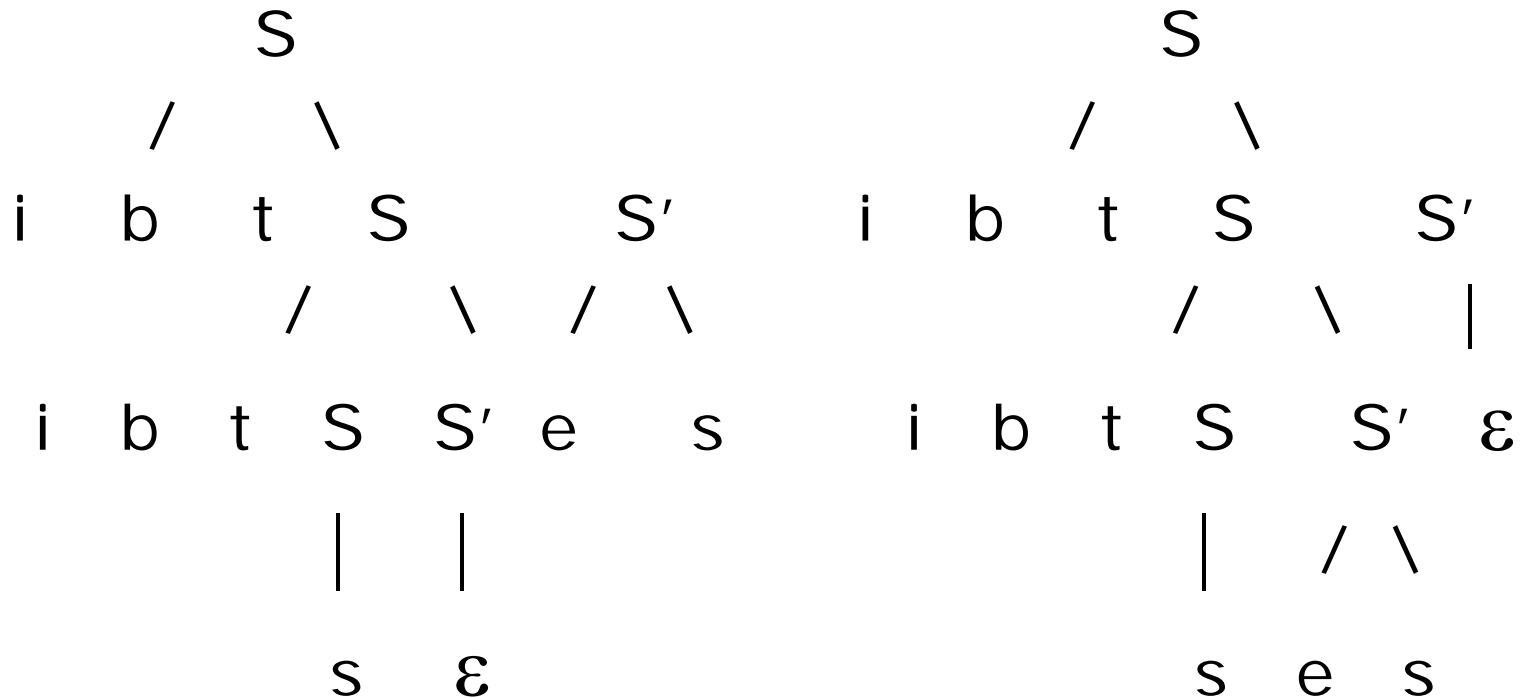
New productions:

$$S \rightarrow i b t S S' \mid s$$

$$S' \rightarrow e S \mid \varepsilon$$

Left Factoring (cont)

Unfortunately, dangling else still exists. There are two parse trees for string: i b t i b t s e s



Bottom-Up SLR(1) Parsing

- SLR(1) is a shift-reduce parser.
- Generate: canonical LR(0) *states* and *items* to enter shift moves into parse table.
- Use follow sets to enter reduces into parse table.

Bottom-Up Grammar

ACC

$E' \rightarrow E$

(1)

$E \rightarrow E + T$

(2)

$E \rightarrow T$

(3)

$T \rightarrow T * F$

(4)

$T \rightarrow F$

(5)

$F \rightarrow (E)$

(6)

$F \rightarrow id$

LR(1) State Table

	id	+	*	()	\$	E	T	F
0	S5			S4			1	2	3
1		S6				ACC			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

Bottom Up Parse

Parse the string \$ id + id * id \$

Scanner Input	Parse Stack
id	0
+	0 id 5
+	0 F 3 (0 and F = 3)
+	0 T 2 (0 and T = 2)
+	0 E 1 (0 and E = 1)
id	0 E 1 + 6
id	0 E 1 + 6 id 5
*	0 E 1 + 6 F 3
*	0 E 1 + 6 T 9
id	0 E 1 + 6 T 9 * 7
\$	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 \Rightarrow ACC

Augmented Productions

Invent a new goal non-terminal symbol and have it go to the old goal symbol.

Example:

Old Productions

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \text{id}$$

Augmented Productions

$$E' \rightarrow E$$
$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \text{id}$$

First/Follow Sets

	First	Follow
E'	id, (\$
E	id, (+, \$,)
T	id, (+, *, \$,)
F	id, (+, *, \$,)

Number Augmented Productions

- ACC
- (1) $E' \rightarrow E$
- (2) $E \rightarrow E + T$
- (3) $E \rightarrow T$
- (4) $T \rightarrow T * F$
- (5) $T \rightarrow F$
- (6) $F \rightarrow (E)$
- (7) $F \rightarrow id$

State I_0

- State I_0 is initially defined by placing a period at the beginning of the right hand side of the augmented production.

Example:

$I_0 \quad E' \rightarrow .E$

Note that $E' \rightarrow .E$ is an item in state I_0 .

Closure

Given the initial items in any state, if the period appears just before a non-terminal, apply closure by adding productions to that state that have the non-terminal on the LHS. Place a period at the beginning of the RHS.

Example:

I_0

- $E' \rightarrow .E$
- $E \rightarrow .E + T$
- $E \rightarrow .T$
- $T \rightarrow .T * F$
- $T \rightarrow .F$
- $F \rightarrow .(E)$
- $F \rightarrow .id$

Closure Example

Example:

I_0	$E' \rightarrow .E$	apply closure
	$E \rightarrow .E + T$	
	$E \rightarrow .T$	apply closure
	$T \rightarrow .T * F$	
	$T \rightarrow .F$	apply closure
	$F \rightarrow .(E)$	
	$F \rightarrow .id$	

Generate New States

New states are generated by moving the period across terminal or non-terminal symbols.

New State Example

In state I_0 , move the period across the E to generate state I_1 .

I_1 $E' \rightarrow E.$

$E \rightarrow E. + T$

Note that there are no opportunities to apply closure in state I_1 so the above represents all of state I_1

Canonical LR(0) States and Items

I_0	$E' \rightarrow .E$	$E \rightarrow .T$
	$E \rightarrow .E + T$	$T \rightarrow .T * F$
	$E \rightarrow .T$	$T \rightarrow .F$
	$T \rightarrow .T * F$	$F \rightarrow .(E)$
	$T \rightarrow .F$	$F \rightarrow .id$
	$F \rightarrow .(E)$	$F \rightarrow id.$
	$F \rightarrow .id$	
I_1	$E' \rightarrow E.$	$E \rightarrow E + .T$
	$E \rightarrow E. + T$	$T \rightarrow .T * F$
		$F \rightarrow .(E)$
I_2	$E \rightarrow T.$	$F \rightarrow .id$
	$T \rightarrow T. * F$	
I_3	$T \rightarrow F.$	$T \rightarrow T * .F$
I_4	$F \rightarrow (.E)$	$F \rightarrow .(E)$
	$E \rightarrow .E + T$	$F \rightarrow .id$

LR(0) States and Items (Cont)

I_8 $F \rightarrow (E .)$
 $E \rightarrow E . + T$

I_9 $E \rightarrow E + T .$
 $T \rightarrow T . * F$

I_{10} $T \rightarrow T * F .$

I_{11} $F \rightarrow (E) .$

LR(1) State Table

	id	+	*	()	\$	E	T	F
0	S5			S4			1	2	3
1		S6				ACC			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

SLR(1) Example 2

ACC

$S' \rightarrow S$

(1)

$S \rightarrow L = R$

(2)

$S \rightarrow R$

(3)

$L \rightarrow *R$

(4)

$L \rightarrow \text{id}$

(5)

$R \rightarrow L$

Ex-2 First/Follow

	First	Follow
S'	id, *	\$
S	id, *	\$
L	id, *	=, \$
R	id, *	=, \$

Ex-2 LR(0) States and Items

I_0	$S' \rightarrow .S$	I_5	$L \rightarrow id.$
	$S \rightarrow .L = R$	I_6	$S \rightarrow L = .R$
	$S \rightarrow .R$		$R \rightarrow .L$
	$L \rightarrow .*R$		$L \rightarrow .*R$
	$L \rightarrow .id$		$L \rightarrow .id$
	$R \rightarrow .L$	I_7	$L \rightarrow * R.$
I_1	$S' \rightarrow S.$	I_8	$R \rightarrow L.$
I_2	$S \rightarrow L . = R$	I_9	$S \rightarrow L = R.$
	$R \rightarrow L.$		
I_3	$S \rightarrow R.$		
I_4	$L \rightarrow * .R$		
	$R \rightarrow .L$		
	$L \rightarrow .*R$		
	$L \rightarrow .id$		

Ex-2, Parse Table

	id	*	=	\$	S	L	R
0	S5	S4			1	2	3
1				ACC			
2			S6/R5	R5			
3				R2			
4	S5	S4				8	7
5			R4	R4			
6	S5	S4				8	9
7			R3	R3			
8			R5	R5			
9				R1			

Ex-2, Shift/Reduce Error

The S6/R5 shift/reduce error means do we:

1. Reduce L to R using the production $R \rightarrow L$,
or
2. Shift = onto the parse stack.

We resolve this conflict in favor of the shift operation because if we reduce L to R, we will have an L followed by an = (R=) and that pattern does not appear in any RHS.

SLR(1) Example 3

ACC

$E' \rightarrow E$

(1)

$E \rightarrow E + E$

(2)

$E \rightarrow E * E$

(3)

$E \rightarrow (E)$

(4)

$E \rightarrow \text{id}$

Ex-3, First/Follow

	First	Follow
E'	id, (\$
E	id, (+, *,), \$

Ex-3 LR(0) States and Items

I_0	$E' \rightarrow \cdot E$		$E \rightarrow \cdot E + E$
	$E \rightarrow \cdot E + E$		$E \rightarrow \cdot E * E$
	$E \rightarrow \cdot E * E$		$E \rightarrow \cdot (E)$
	$E \rightarrow \cdot (E)$		$E \rightarrow \cdot id$
	$E \rightarrow \cdot id$	I_5	$E \rightarrow E * \cdot E$
I_1	$E' \rightarrow E \cdot$		$E \rightarrow \cdot E + E$
	$E \rightarrow E \cdot + E$		$E \rightarrow \cdot E * E$
	$E \rightarrow E \cdot * E$		$E \rightarrow \cdot (E)$
I_2	$E \rightarrow (\cdot E)$		$E \rightarrow \cdot id$
	$E \rightarrow \cdot E + E$	I_6	$E \rightarrow (E \cdot)$
	$E \rightarrow \cdot E * E$		$E \rightarrow E \cdot + E$
	$E \rightarrow \cdot (E)$		$E \rightarrow E \cdot * E$
	$E \rightarrow \cdot id$	I_7	$E \rightarrow E + E \cdot$
I_3	$E \rightarrow id \cdot$		$E \rightarrow E \cdot + E$
I_4	$E \rightarrow E + \cdot E$		$E \rightarrow E \cdot * E$

Ex-3, LR(0) States/Items (cont)

I_8 $E \rightarrow E * E.$
 $E \rightarrow E . + E$
 $E \rightarrow E . * E$

I_9 $E \rightarrow (E).$

Ex-3, Parse Table

	id	+	*	()	\$	E
0	S3			S2			1
1		S4	S5			ACC	
2	S3			S2			6
3		R4	R4		R4	R4	
4	S3			S2			7
5	S3			S2			8
6		S4	S5		S9		
7		R1/S4	R1/S5		R1	R1	
8		R2/S4	R2/S5		R2	R2	
9		R3	R3		R3	R3	

Resolving Conflicts

Resolve in favor of:

R1/S4 \Rightarrow E + E. +

R1 associativity

R1/S5 \Rightarrow E + E. *

S5 precedence

R2/S4 \Rightarrow E * E. +

R2 precedence

R2/S5 \Rightarrow E * E. *

R2 associativity

LR(1) Parsing

$S \rightarrow CC$

$C \rightarrow cC \mid d$

ACC

$S' \rightarrow S$

(1)

$S \rightarrow CC$

(2)

$C \rightarrow cC$

(3)

$C \rightarrow d$

LR(1), First/Follow

	First	Follow
S'	c,d	\$
S	c,d	\$
C	c,d	\$,c,d

Initial State I_0

$I_0 \quad S' \rightarrow .S \{ \$ \}$

Each item has a look-ahead set ($\{ \$ \}$ in this case).

LR(1), Closure

Each time closure is applied:

1. Designate the string of (possibly empty) terminals and non-terminals following the non-terminal you applied closure to as α .
2. Invent a non-terminal L and assume L has the current look-ahead set as its first set.
3. The look-ahead set of the items added when applying closure is $\text{first}(\alpha L)$.

LR(1), New States

New states are generated similar to the way they are with LR(0) items, by moving the period across terminals or non-terminals, only with LR(1) items the look-ahead set for that item before the period is moved becomes the look-ahead set for the item in the new state after the period is moved.

LR(1) States and Items

I_0	$S' \rightarrow .S$ { \$ }	I_4	$C \rightarrow d.$ { c, d }
	$S \rightarrow .CC$ { \$ }	I_5	$S \rightarrow CC.$ { \$ }
	$C \rightarrow .cC$ { c, d }	I_6	$C \rightarrow c.C$ { \$ }
	$C \rightarrow .d$ { c, d }		$C \rightarrow .cC$ { \$ }
I_1	$S' \rightarrow S.$ { \$ }		$C \rightarrow .d$ { \$ }
I_2	$S \rightarrow C.C$ { \$ }	I_7	$C \rightarrow d.$ { \$ }
	$C \rightarrow .cC$ { \$ }	I_8	$C \rightarrow cC.$ { c, d }
	$C \rightarrow .d$ { \$ }	I_9	$C \rightarrow cC.$ { \$ }
I_3	$C \rightarrow c.C$ { c, d }		
	$C \rightarrow .cC$ { c, d }		
	$C \rightarrow .d$ { c, d }		

LR(1), Reduce Entries

Instead of placing reduces in parse table according to the follow set of the non-terminal on the left-hand side, use the look-ahead set to enter reduces into the table.

LR(1), Parse Table

	c	d	\$	S	C
0	S3	S4		1	2
1			ACC		
2	S6	S7			5
3	S3	S4			8
4	R3	R3			
5			R1		
6	S6	S7			9
7			R3		
8	R2	R2			
9			R2		

LR(1)/SLR(1), Comparison

LR(1) recognizes a wider range of languages

LR(1) parse tables are huge. For a typical computer language, SLR(1) would contain a few hundred states, whereas LR(1) would contain a few thousand states.

LALR(1)

LALR(1) – Look-ahead LR(1)

Notice that the following states have the same items but different look-ahead sets for at least one of the items:

1. 3 and 6 Combine into new state 36
2. 4 and 7 Combine into new state 47
3. 8 and 9 Combine into new state 89

LALR(1), Parse Table

	c	d	\$	S	C
0	S36	S47		1	2
1			ACC		
2	S36	S47			5
36	S36	S47			89
47	R3	R3	R3		
5			R1		
89	R2	R2	R2		

Reporting Errors

Notice that the grammar:

- ACC
- $S' \rightarrow S$
- (1) $S \rightarrow CC$
- (2) $C \rightarrow cC$
- (3) $C \rightarrow d$

Recognizes sentences that can be described by the regular expression

$c^* d c^* d$

LR(1) vs LALR(1) Error Reporting

The string `ccd` is not in the grammar.
Compare how many steps LR(1) takes to detect the error in comparison with LALR(1).

LR(1), Error Reporting

c 0

c 0c3

d 0c3c3

\$ 0c3c3d4 –

Error no entry for \$ in state 4

LALR(1), Error Reporting

c 0

c 0c36

d 0c36c36

\$ 0c36c36d47

\$ 0c36c36C89

\$ 0c36C89

\$ 0C2 – Error no entry for \$ in state 2

LR(1), Example 2

ACC

$E' \rightarrow E$

(1)

$E \rightarrow E + T$

(2)

$E \rightarrow T$

(3)

$T \rightarrow (E)$

(4)

$T \rightarrow \text{id}$

Ex 2, First/Follow

E'	(, id	\$
E	(, id	\$,), +
T	(, id	\$,), +

Ex 2, LR(1) States and Items

I_0	$E' \rightarrow .E$	$\{\$, +\}$	I_4	$T \rightarrow id.$	$\{\$, +\}$
	$E \rightarrow .E + T$	$\{\$, +\}$	I_5	$E \rightarrow E + .T$	$\{\$, +\}$
	$E \rightarrow .T$	$\{\$, +\}$		$T \rightarrow .(E)$	$\{\$, +\}$
	$T \rightarrow .(E)$	$\{\$, +\}$		$T \rightarrow .id$	$\{\$, +\}$
	$T \rightarrow .id$	$\{\$, +\}$	I_6	$T \rightarrow (E .)$	$\{\$, +\}$
I_1	$E' \rightarrow E.$	$\{\$\}$		$E \rightarrow E. + T$	$\{\}, +\}$
	$E \rightarrow E. + T$	$\{\$, +\}$	I_7	$E \rightarrow T.$	$\{\}, +\}$
I_2	$E \rightarrow T.$	$\{\$, +\}$	I_8	$T \rightarrow (.E)$	$\{\}, +\}$
I_3	$T \rightarrow (.E)$	$\{\$, +\}$		$E \rightarrow .E + T$	$\{\}, +\}$
	$E \rightarrow .E + T$	$\{\}, +\}$		$E \rightarrow .T$	$\{\}, +\}$
	$E \rightarrow .T$	$\{\}, +\}$		$T \rightarrow .(E)$	$\{\}, +\}$
	$T \rightarrow .(E)$	$\{\}, +\}$		$T \rightarrow .id$	$\{\}, +\}$
	$T \rightarrow .id$	$\{\}, +\}$	I_9	$T \rightarrow id.$	$\{\}, +\}$

Ex 2, (cont)

I_{10} $E \rightarrow E + T. \{ \$, + \}$

I_{11} $T \rightarrow (E). \{ \$, + \}$

I_{12} $E \rightarrow E + .T \{), + \}$

$T \rightarrow .(E) \{), + \}$

$T \rightarrow .id \{), + \}$

I_{13} $T \rightarrow (E .) \{), + \}$

$E \rightarrow E. + T \{), + \}$

I_{14} $E \rightarrow E + T. \{), + \}$

I_{15} $T \rightarrow (E). \{), + \}$

Similar States:

$2/7, 3/8, 4/9, 5/12, 6/13, 10/14, 11/15$

Ex 2, LR(1) Parse Table

	id	+	()	\$	E	T
0	S4		S3			1	2
1		S5			ACC		
2		R2			R2		
3	S9		S8			6	7
4		R4			R4		
5	S4		S3				10
6		S12		S11			
7		R2		R2			
8	S9		S8			13	7
9		R4		R4			
10		R1			R1		
11		R3			R3		
12	S9		S8				14
13		S12		S15			
14		R1		R1			
15		R3		R3			

Ex 2, LALR(1) Parse Table

	id	+	()	\$	E	T
0	S4(9)		S3(8)			1	2(7)
1		S5(12)			ACC		
2(7)		R2		R2	R2		
3(8)	S4(9)		S3(8)			6(13)	2(7)
4(9)		R4		R4	R4		
5(12)	S4(9)		S3(8)				10(14)
6(13)		S5(12)		S11(15)			
10(14)		R1		R1	R1		
11(15)		R3		R3	R3		

4.35, Augmented Productions

ACC

$$E' \rightarrow E$$

$$(1) \quad E \rightarrow E + T$$

$$(2) \quad E \rightarrow T$$

$$(3) \quad T \rightarrow T F$$

$$(4) \quad T \rightarrow F$$

$$(5) \quad F \rightarrow F *$$

$$(6) \quad F \rightarrow a$$

$$(7) \quad F \rightarrow b$$

4.35, First/Follow

	First	Follow
E'	a, b	\$
E	a, b	+, \$
T	a, b	a, b, +, \$
F	a, b	*, a, b, +, \$

4.35, LR(0) States/Items

I_0 $E' \rightarrow .E$
 $E \rightarrow .E + T$
 $E \rightarrow .T$
 $T \rightarrow .T F$
 $T \rightarrow .F$
 $F \rightarrow .F *$
 $F \rightarrow .a$
 $F \rightarrow .b$

I_1 $E' \rightarrow E.$
 $E \rightarrow E. + T$

I_2 $E \rightarrow T.$
 $T \rightarrow T.F$
 $F \rightarrow .F *$
 $F \rightarrow .a$
 $F \rightarrow .b$

I_3 $T \rightarrow F.$
 $F \rightarrow F.*$

I_4 $F \rightarrow a.$

I_5 $F \rightarrow b.$

I_6 $E \rightarrow E + .T$
 $T \rightarrow .T F$

$T \rightarrow .F$

$F \rightarrow .F *$

$F \rightarrow .a$

$F \rightarrow .b$

I_7 $T \rightarrow T F.$
 $F \rightarrow F.*$

I_8 $F \rightarrow F*.$

4.35, LR(0) States/Items

I_9 $E \rightarrow E + T.$
 $T \rightarrow T .F$
 $F \rightarrow .F *$
 $F \rightarrow .a$
 $F \rightarrow .b$

4.35, Parse Table

	a	b	+	*	\$	E	T	F
0	S4	S5				1	2	3
1			S6		ACC			
2	S4	S5	R2		R2			7
3	R4	R4	R4	S8	R4			
4	R6	R6	R6	R6	R6			
5	R7	R7	R7	R7	R7			
6	S4	S5					9	7
7	R3	R3	R3	S8	R3			
8	R5	R5	R5	R5	R5			
9	S4	S5	R1		R1			7

4.39, Augmented Productions

ACC

$$S' \rightarrow S$$

(1)

$$S \rightarrow A a$$

(2)

$$S \rightarrow b A c$$

(3)

$$S \rightarrow d c$$

(4)

$$S \rightarrow b d a$$

(5)

$$A \rightarrow d$$

4.39, First/Follow

	First	Follow
S'	b, d	\$
S	b, d	\$
A	d	a, c

4.39 States/Items

I_0	$S' \rightarrow .S$	I_4	$S \rightarrow d .c$
	$S \rightarrow .A a$		$A \rightarrow d.$
	$S \rightarrow .b A c$	I_5	$S \rightarrow A a.$
	$S \rightarrow .d c$	I_6	$S \rightarrow b A .c$
	$S \rightarrow .b d a$	I_7	$S \rightarrow b d .a$
	$A \rightarrow .d$		$A \rightarrow d.$
I_1	$S' \rightarrow S.$	I_8	$S \rightarrow d c.$
I_2	$S \rightarrow A .a$	I_9	$S \rightarrow b A c.$
I_3	$S \rightarrow b .A c$	I_{10}	$S \rightarrow b d a.$
	$S \rightarrow b .d a$		
	$A \rightarrow .d$		

4.39, State Table

	a	b	c	d	\$	S	A
0		S3		S4		1	2
1					ACC		
2	S5						
3				S7			6
4	R5		S8/R5				
5					R1		
6			S9				
7	S11/R5		R5				
8					R3		
9					R2		
10					R4		

4.40, Augmented Productions

ACC

$S' \rightarrow S$

(1)

$S \rightarrow A a$

(2)

$S \rightarrow b A c$

(3)

$S \rightarrow B c$

(4)

$S \rightarrow b B a$

(5)

$A \rightarrow d$

(6)

$B \rightarrow d$

4.40, First/Follow

	First	Follow
S'	b, d	\$
S	b, d	\$
A	d	a, c
B	d	a, c

4.40, LR(1) States/Items

I_0	$S' \rightarrow .S$	$\{\$ \}$	I_4	$S \rightarrow B .c$	$\{\$ \}$
	$S \rightarrow .A a$	$\{\$ \}$	I_5	$A \rightarrow d.$	$\{a \}$
	$S \rightarrow .b A c$	$\{\$ \}$		$B \rightarrow d.$	$\{c \}$
	$S \rightarrow .B c$	$\{\$ \}$	I_6	$S \rightarrow A a.$	$\{\$ \}$
	$S \rightarrow .b B a$	$\{\$ \}$	I_7	$S \rightarrow b A .c$	$\{\$ \}$
	$A \rightarrow .d$	$\{a \}$	I_8	$S \rightarrow b B .a$	$\{\$ \}$
	$B \rightarrow .d$	$\{c \}$	I_9	$A \rightarrow d.$	$\{c \}$
I_1	$S' \rightarrow S.$	$\{\$ \}$		$B \rightarrow d.$	$\{a \}$
I_2	$S \rightarrow A .a$	$\{\$ \}$	I_{10}	$S \rightarrow B c.$	$\{\$ \}$
I_3	$S \rightarrow b .A c$	$\{\$ \}$	I_{11}	$S \rightarrow b A c.$	$\{\$ \}$
	$S \rightarrow b .B a$	$\{\$ \}$	I_{12}	$S \rightarrow b B a.$	$\{\$ \}$
	$A \rightarrow .d$	$\{c \}$			
	$B \rightarrow .d$	$\{a \}$			

Error Detection and Recovery

ACC	$E' \rightarrow E$
(1)	$E \rightarrow E + E$
(2)	$E \rightarrow E * E$
(3)	$E \rightarrow (E)$
(4)	$E \rightarrow id$

Ex-3, Parse Table

	id	+	*	()	\$	E
0	S3	E1	E1	S2	E2	E1	1
1	E3	S4	S5	E3	E2	ACC	
2	S3	E1	E1	S2	E2	E1	6
3	R4	R4	R4	R4	R4	R4	
4	S3	E1	E1	S2	E2	E1	7
5	S3	E1	E1	S2	E2	E1	8
6	E3	S4	S5	E3	S9	E4	
7	R1	R1	S5	R1	R1	R1	
8	R2	R2	R2	R2	R2	R2	
9	R3	R3	R3	R3	R3	R3	

Error Conditions

- E1 – Entered from states 0, 2, 4, 5 when operand or (is expected but an operator or \$ is found instead. Push id on stack and cover with 3. Issue diagnostic “Missing operand”.
- E2 – Entered from states 0, 1, 2, 4, 5 when unexpected) is encountered. Remove) from input string and issue diagnostic “Unexpected right parenthesis”.
- E3 – Entered from states 1, 6 when operator is expected but operand or) is found instead. Push + on stack and cover with 4. Issue diagnostic “Missing operator”
- E4 – Entered form state 6 when operator or) is expected but \$ is found instead. Push) on stack and cover with 9. Issue diagnostic “Unbalanced parenthesis”.

Example 1

\$ id +) \$

Input

Parse Stack

id

0

id

0 id 3

+

0 E 1

)

0 E 1 + 4

\$

0 E 1 + 4 "Unexpected right parenthesis"

\$

0 E 1 + 4 id 3 "Missing operand"

\$

0 E 1 + 4 E 7

\$

0 E 1 Accept

Example 2

\$ (id₁ id₂ \$

Input	Parse Stack
(0
id ₁	0 (2
id ₂	0 (2 id ₁ 3
id ₂	0 (2 E 6
id ₂	0 (2 E 6 + 4 “Missing operator”
\$	0 (2 E 6 + 4 id ₂ 3
\$	0 (2 E 6 + 4 E 7
\$	0 (2 E 6
\$	0 (2 E 6) 9 “Unbalanced Parenthesis”
\$	0 E 1 Accept

Chomsky Normal Form (CNF)

- Grammar must be Context Free
- All productions are of the form:
 - $A \rightarrow a$ RHS is a terminal
 - $A \rightarrow BC$ RHS is two non-terminals
- If ε (empty string) is in language and $S \rightarrow \varepsilon$, S never appears on RHS of any production.

What is Normal Form?

- A Grammar in normal form is unique.
- There are no two different normal forms.
- In order to determine if two grammars are equivalent, reduce them both to normal form.
- With a few simple transformations such as NT name changes the normal form productions should be the same if they are equivalent.

CNF, Transformation

$$G_1 = (N_1, \Sigma, S, P_1) \Rightarrow G_2 = (N_2, \Sigma, S, P_2)$$

G_1 = Original Context Free Grammar

G_2 = CNF Grammar

N_1 = Original set of non-terminals

N_2 = Set of CNF non-terminals

Σ = Set of terminal symbols

P_1 = Original Set of Productions

P_2 = Set of CNF productions

CNF Algorithm

1. Add all productions in P_1 of the form:

$A \rightarrow a$

$A \rightarrow BC$

to P_2

2. For each production of the form:

$A \rightarrow X_1 X_2 \dots X_n$

add to P_2

$A \rightarrow X_1' \langle X_2 \dots X_n \rangle$

$\langle X_2 \dots X_n \rangle \rightarrow X_2' \langle X_3 \dots X_n \rangle$

$\langle X_{n-1} X_n \rangle \rightarrow X_{n-1}' X_n'$

3. If $X_i \in N$, then leave as X_i .

If $X_i \in \Sigma$, then rewrite it as new non-terminal X_i' and add new production $X_i' \rightarrow X_i$ (remember that X_i is a terminal)

CNF, Ex 1

$P_1 =$

$A \rightarrow bCDeF$

$P_2 =$

$A \rightarrow X_1' \langle CDeF \rangle$

$X_1' \rightarrow b$

$\langle CDeF \rangle \rightarrow C \langle DeF \rangle$

$\langle DeF \rangle \rightarrow D \langle eF \rangle$

$\langle eF \rangle \rightarrow X_2' F$

$X_2' \rightarrow e$

Two NTs on RHS

Single terminal on RHS

Two NTs on RHS

Two NTs on RHS

Two NTs on RHS

Single terminal on RHS

CNF, Ex 2

$G_1 =$ $S \rightarrow aAB \mid BA$

$A \rightarrow BbB \mid a$

$B \rightarrow AS \mid b$

$G_2 =$ $S \rightarrow BA$

$A \rightarrow a$

$B \rightarrow AS \mid b$

$S \rightarrow X_1' \langle AB \rangle$

$X_1' \rightarrow a$

$\langle AB \rangle \rightarrow AB$

$A \rightarrow B \langle bB \rangle$

$\langle bB \rangle \rightarrow X_2' B$

$X_2' \rightarrow b$

Greibach Normal Form

- Productions have the form:

$S \rightarrow \varepsilon$ S is the goal NT

$A \rightarrow b$ b is a terminal symbol

$A \rightarrow b \alpha$ α is a string of NTs

GNF, Example

$$S \rightarrow AB$$

$$A \rightarrow aAb \mid \varepsilon$$

$$B \rightarrow Bb \mid \varepsilon$$

Greibach Normal Form (derived without showing work)

$$S \rightarrow \varepsilon$$

$$B \rightarrow bC_3$$

$$S \rightarrow aAC_4C_1$$

$$B \rightarrow b$$

$$S \rightarrow aC_4C_1$$

$$C_1 \rightarrow bC_3$$

$$S \rightarrow bC_2$$

$$C_2 \rightarrow bC_2$$

$$S \rightarrow aC_4$$

$$C_2 \rightarrow b$$

$$S \rightarrow aAC_4$$

$$C_3 \rightarrow bC_3$$

$$S \rightarrow b$$

$$C_3 \rightarrow b$$

$$A \rightarrow aAC_4$$

$$C_4 \rightarrow b$$

$$A \rightarrow aC_4$$

LALR(1) Parsing

- Parse tables are the same size as SLR(1) parsing.
- Recognizes more context-free grammars than SLR(1) – less likely to generate shift/reduce conflicts than SLR(1).

LALR(1), Grammar

Grammar

- ACC
- (1) $S' \rightarrow S$
- (2) $S \rightarrow L = R$
- (3) $S \rightarrow R$
- (4) $L \rightarrow *R$
- (5) $L \rightarrow \text{id}$
- (6) $R \rightarrow L$

LALR(1), First Follow

	First	Follow
S'	$*, id$	$\$$
S	$*, id$	$\$$
L	$*, id$	$\$, =$
R	$*, id$	$\$, =$

LALR(1), Kernel Items

- Generate LR(0) states and items in the same manner as you did when doing an SLR(1) parse.
- Kernel items are those items generated when generating LR(0) items that are not added as a result of applying closure.
- The kernel items in the next slide are shown in green.

LALR(1), LR(0) States and Items

Gray are kernel items; black are non-kernel items.

I_0	$S' \rightarrow .S$	I_5	$L \rightarrow id.$
	$S \rightarrow .L = R$	I_6	$S \rightarrow L = .R$
	$S \rightarrow .R$		$R \rightarrow .L$
	$L \rightarrow .*R$		$L \rightarrow .*R$
	$L \rightarrow .id$		$L \rightarrow .id$
	$R \rightarrow .L$	I_7	$L \rightarrow * R.$
I_1	$S' \rightarrow S.$	I_8	$R \rightarrow L.$
I_2	$S \rightarrow L . = R$	I_9	$S \rightarrow L = R.$
	$R \rightarrow L.$		
I_3	$S \rightarrow R.$		
I_4	$L \rightarrow * .R$		
	$R \rightarrow .L$		
	$L \rightarrow .*R$		
	$L \rightarrow .id$		

LALR(1), General Propagate Set-

- Whereas LR(1) shows every step with regard to look-ahead generation, LALR(1) uses some shortcuts.
- Assume that # is a set of look-ahead symbols that represent the look-ahead set of a kernel item.
- If period is before a terminal symbol, just list the kernel item's general look-ahead set.
- If period is before a non-terminal, apply closure wherever possible to see how # is propagated.

LALR(1), Kernel Closure

I_0 $S' \rightarrow .S$ $\{\#_1\}$
 $S \rightarrow .L = R$ $\{\#_1\}$
 $S \rightarrow .R$ $\{\#_1\}$
 $L \rightarrow .*R$ $\{=, \#_1\}$
 $L \rightarrow .id$ $\{=, \#_1\}$
 $R \rightarrow .L$ $\{\#_1\}$

I_2 $S \rightarrow L . = R$ $\{\#_2\}$

← Period is before a terminal

I_4 $L \rightarrow * .R$ $\{\#_3\}$
 $R \rightarrow .L$ $\{\#_3\}$
 $L \rightarrow .*R$ $\{\#_3\}$
 $L \rightarrow .id$ $\{\#_3\}$

I_6 $S \rightarrow L = .R$ $\{\#_4\}$
 $R \rightarrow .L$ $\{\#_4\}$
 $L \rightarrow .*R$ $\{\#_4\}$
 $L \rightarrow .id$ $\{\#_4\}$

LALR(1), Propagate Table

- What look-ahead set does a new state get when you pass the period across a terminal or non-terminal in a kernel state.
- Propagate table contains entries on “from” side for all kernel items that have a period before a terminal symbol or non-terminal symbol.
- Entries on “to” side are all of the states the “from” look-ahead set is passed to.

LALR(1), Propagate Table

From	To
$I_0 \quad S' \rightarrow \cdot S$	$I_1 \quad S' \rightarrow S \cdot$ $I_2 \quad S \rightarrow L \cdot = R$ $I_2 \quad R \rightarrow L \cdot$ $I_3 \quad S \rightarrow R \cdot$ $I_4 \quad L \rightarrow \cdot * R$ $I_5 \quad L \rightarrow id \cdot$
$I_2 \quad S \rightarrow L \cdot = R$	$I_6 \quad S \rightarrow L = \cdot R$
$I_4 \quad L \rightarrow \cdot * R$	$I_4 \quad L \rightarrow \cdot * R$ $I_5 \quad L \rightarrow id \cdot$ $I_7 \quad L \rightarrow * R \cdot$ $I_8 \quad R \rightarrow L \cdot$
$I_6 \quad S \rightarrow L = \cdot R$	$I_4 \quad L \rightarrow \cdot * R$ $I_5 \quad L \rightarrow id \cdot$ $I_8 \quad R \rightarrow L \cdot$ $I_9 \quad S \rightarrow L = R \cdot$

LALR(1), Pass Table

	Init	Pass 1	Pass 2	Pass 3
I_0 $S' \rightarrow .S$	\$	\$	\$	\$
I_1 $S' \rightarrow S.$		\$	\$	\$
I_2 $S \rightarrow L . = R$		\$	\$	\$
I_2 $R \rightarrow L.$		\$	\$	\$
I_3 $S \rightarrow R.$		\$	\$	\$
I_4 $L \rightarrow * .R$	=	=\$	=\$	=\$
I_5 $L \rightarrow id.$	=	=\$	=\$	=\$
I_6 $S \rightarrow L = .R$			\$	\$
I_7 $L \rightarrow * R.$		=	=\$	=\$
I_8 $R \rightarrow L.$		=	=\$	=\$
I_9 $S \rightarrow L = R.$				\$

For each pass, black is old and grey is new

LALR(1), Parse Table

	id	*	=	\$	S	L	R
0	S5	S4			1	2	3
1				ACC			
2			S6	R5			
3				R2			
4	S5	S4				8	7
5			R4	R4			
6	S5	S4				8	9
7			R3	R3			
8			R5	R5			
9				R1			

Example 2

ACC

$E' \rightarrow E$

(1)

$E \rightarrow E + T$

(2)

$E \rightarrow T$

(3)

$T \rightarrow T * F$

(4)

$T \rightarrow F$

(5)

$F \rightarrow (E)$

(6)

$F \rightarrow \text{id}$

Ex 2, First/Follow Sets

	First	Follow
E'	id, (\$
E	id, (+, \$,)
T	id, (+, *, \$,)
F	id, (+, *, \$,)

Ex 2, LR(0) States and Items

Grey are kernel items; black are non-kernel items.

I_0	$E' \rightarrow .E$	$E \rightarrow .T$
	$E \rightarrow .E + T$	$T \rightarrow .T * F$
	$E \rightarrow .T$	$T \rightarrow .F$
	$T \rightarrow .T * F$	$F \rightarrow .(E)$
	$T \rightarrow .F$	$F \rightarrow .id$
	$F \rightarrow .(E)$	$F \rightarrow id.$
	$F \rightarrow .id$	
I_1	$E' \rightarrow E.$	$E \rightarrow E + .T$
	$E \rightarrow E. + T$	$T \rightarrow .T * F$
		$F \rightarrow .(E)$
I_2	$E \rightarrow T.$	$F \rightarrow .id$
	$T \rightarrow T. * F$	
		$T \rightarrow T * .F$
I_3	$T \rightarrow F.$	$F \rightarrow .(E)$
I_4	$F \rightarrow (.E)$	$F \rightarrow .id$
	$E \rightarrow .E + T$	

Ex 2, LR(0) States and Items (Cont)

I_8 $F \rightarrow (E \cdot)$
 $E \rightarrow E \cdot + T$

I_9 $E \rightarrow E + T \cdot$
 $T \rightarrow T \cdot * F$

I_{10} $T \rightarrow T * F \cdot$

I_{11} $F \rightarrow (E) \cdot$

Ex 2, Passing

I_0	$E' \rightarrow .E$	$\{\#_1\}$
	$E \rightarrow .E + T$	$\{\#_1, +\}$
	$E \rightarrow .T$	$\{\#_1, +\}$
	$T \rightarrow .T * F$	$\{\#_1, +, *\}$
	$T \rightarrow .F$	$\{\#_1, +, *\}$
	$F \rightarrow .(E)$	$\{\#_1, +, *\}$
	$F \rightarrow .id$	$\{\#_1, +, *\}$
I_1	$E \rightarrow E .+ T$	$\{\#_2\}$
I_2	$T \rightarrow T . * F$	$\{\#_3\}$
I_4	$F \rightarrow (.E)$	$\{\#_4\}$
	$E \rightarrow .E + T$	$\{), +\}$
	$E \rightarrow .T$	$\{), +\}$
	$T \rightarrow .T * F$	$\{), +, *\}$
	$T \rightarrow .F$	$\{), +, *\}$
	$F \rightarrow .(E)$	$\{), +, *\}$
	$F \rightarrow .id$	$\{), +, *\}$

Ex 2, Passing # (cont)

I_6	$E \rightarrow E + \cdot T$	$\{\#_5\}$
	$T \rightarrow \cdot T * F$	$\{\#_5, *\}$
	$T \rightarrow \cdot F$	$\{\#_5, *\}$
	$F \rightarrow \cdot (E)$	$\{\#_5, *\}$
	$F \rightarrow \cdot id$	$\{\#_5, *\}$
I_7	$T \rightarrow T * \cdot F$	$\{\#_6\}$
	$F \rightarrow \cdot (E)$	$\{\#_6\}$
	$F \rightarrow \cdot id$	$\{\#_6\}$
I_8	$F \rightarrow (E \cdot)$	$\{\#_7\}$
I_8	$E \rightarrow E \cdot + T$	$\{\#_8\}$
I_9	$T \rightarrow T \cdot * F$	$\{\#_9\}$

Ex 2, Propagate Table

From	To
$I_0 \quad E' \rightarrow \cdot E$	$I_1 \quad E' \rightarrow E \cdot$ $I_1 \quad E \rightarrow E \cdot + T$ $I_2 \quad E \rightarrow T \cdot$ $I_2 \quad T \rightarrow T \cdot * L$ $I_3 \quad T \rightarrow F \cdot$ $I_4 \quad F \rightarrow (\cdot E)$ $I_5 \quad F \rightarrow id \cdot$
$I_1 \quad E \rightarrow E \cdot + T$	$I_6 \quad E \rightarrow E + \cdot T$
$I_2 \quad T \rightarrow T \cdot * F$	$I_7 \quad T \rightarrow T * \cdot F$
$I_4 \quad F \rightarrow (\cdot E)$	$I_8 \quad F \rightarrow (E \cdot)$

LALR(1), Propagate Table

From	To
$I_6 \ E \rightarrow E + \cdot T$	$I_9 \ E \rightarrow E + T \cdot$ $I_9 \ T \rightarrow T \cdot * F$ $I_3 \ T \rightarrow F \cdot$ $I_4 \ F \rightarrow (\cdot E)$ $I_5 \ F \rightarrow id \cdot$
$I_7 \ T \rightarrow T * \cdot F$	$I_{10} \ T \rightarrow T * F \cdot$ $I_4 \ F \rightarrow (\cdot E)$ $I_5 \ F \rightarrow id \cdot$
$I_8 \ F \rightarrow (E \cdot)$	$I_{11} \ F \rightarrow (E) \cdot$
$I_8 \ E \rightarrow E \cdot + T$	$I_6 \ E \rightarrow E + \cdot T$
$I_9 \ T \rightarrow T \cdot * F$	$I_7 \ T \rightarrow T * \cdot F$

Ex 2, Pass Table

	Init	Pass 1	Pass 2	Pass 3	Pass 4
I_0 $E' \rightarrow .E$	\$	\$	\$	\$	\$
I_1 $E' \rightarrow E.$		\$	\$	\$	\$
I_1 $E \rightarrow E .+ T$	+	+, \$	+, \$	+, \$	+, \$
I_2 $E \rightarrow T.$	+,)	+,), \$	+,), \$	+,), \$	+,), \$
I_2 $T \rightarrow T .* F$	+, *,)	+, *,), \$	+, *,), \$	+, *,), \$	+, *,), \$
I_3 $T \rightarrow F.$	+, *,)	+, *,), \$	+, *,), \$	+, *,), \$	+, *,), \$
I_4 $F \rightarrow (.E)$	+, *,)	+, *,), \$	+, *,), \$	+, *,), \$	+, *,), \$
I_5 $F \rightarrow id.$	+, *,)	+, *,), \$	+, *,), \$	+, *,), \$	+, *,), \$
I_6 $E \rightarrow E + .T$		+,)	+,), \$	+,), \$	+,), \$
I_7 $T \rightarrow T * .F$		+, *,)	+, *,), \$	+, *,), \$	+, *,), \$

For each pass, black is old and grey is new

Ex 2, Pass Table (cont)

	Init	Pass 1	Pass 2	Pass 3	Pass 4
$I_8 \quad F \rightarrow (E .)$		$+, *,)$	$+, *,), \$$	$+, *,), \$$	$+, *,), \$$
$I_8 \quad E \rightarrow E . + T$	$), +$	$), +$	$), +$	$), +$	$), +$
$I_9 \quad E \rightarrow E + T .$			$+,)$	$+,), \$$	$+,), \$$
$I_9 \quad T \rightarrow T . * F$	$*$	$*$	$*, +,)$	$*, +,), \$$	$*, +,), \$$
$I_{10} \quad T \rightarrow T * F .$			$+, *,)$	$+, *,), \$$	$+, *,), \$$
$I_{11} \quad F \rightarrow (E) .$			$+, *,)$	$+, *,), \$$	$+, *,), \$$

LL(k)/LR(k) Grammars

The grammar:

$$A \rightarrow B a b$$
$$B \rightarrow a \mid \varepsilon$$

Produces the language $a a b$ and $a b$. If you were to recognize the sentence $a a b$, the scanner would first return the first symbol a and the parser would know to apply the production:

$$A \rightarrow B a b$$

Then, the parser would next seek to expand the non-terminal B , again based only on the knowledge of the symbol a . Notice that applying either of the B productions will lead to a match of a , so we can't make the B parsing decision with a look-ahead of just 1.

Look-Ahead of 2

The parser can require that the scanner get 2 tokens instead of one each time it needs to make a parsing decision. In this case, after the production:

$$A \rightarrow B a b$$

is applied, the parser uses a a to make it's parsing decision. Now,

$$B \rightarrow \varepsilon$$

fails to meet the look-ahead requirements, whereas

$$B \rightarrow a$$

does meet the requirements and is chosen by the parser.

This grammar is LL(2)/LR(2).

Push-Down Automata

- Equivalent to Context-Free Grammars
- Context-Free Grammars can “remember”. For example, it is possible to represent balanced parentheses with a Context-Free Grammar. It is NOT possible to represent balanced parentheses with a Regular Grammar, Regular Expression, or Finite Automata.
- Push-down automata maintains a stack.

Balanced Parentheses

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

is a grammar that recognizes balanced parentheses.

PDA Moves

$$\delta(\text{state}_1, \text{symbol}_1, \text{symbol}_2) = (\text{state}_2, \text{symbol}_3)$$

Where:

state_1 is the state before the δ move.

state_2 is the state after the δ move.

symbol_1 is the next input symbol.

symbol_2 is the symbol to be replaced on the PDA stack.

symbol_3 replaces symbol_2 at the top of the PDA stack after the move

CFG to PDA

For every production $A \rightarrow \alpha$ add the move:

$$\delta(q, \varepsilon, A) = (q, \alpha)$$

For every terminal symbol t add the move:

$$\delta(q, t, t) = (q, \varepsilon)$$

PDA Example

The grammar:

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow (E) \mid \text{id}$$

Produces the PDA δ -moves:

1. $\delta(q, \varepsilon, E) = (q, E + T) \text{ or } (q, T)$
2. $\delta(q, \varepsilon, T) = (q, T * F) \text{ or } (q, F)$
3. $\delta(q, \varepsilon, F) = (q, (E)) \text{ or } (q, \text{id})$
4. $\delta(q, +, +) = (q, \varepsilon)$
5. $\delta(q, *, *) = (q, \varepsilon)$
6. $\delta(q, (, () = (q, \varepsilon)$
7. $\delta(q,),)) = (q, \varepsilon)$
8. $\delta(q, \text{id}, \text{id}) = (q, \varepsilon)$

PDA example (cont)

$(q, id + id, E) \Rightarrow$

rule 1 first choice

$(q, id + id, E + T) \Rightarrow$

rule 1 second choice

$(q, id + id, T + T) \Rightarrow$

rule 2 second choice

$(q, id + id, F + T) \Rightarrow$

rule 3 second choice

$(q, id + id, id + T) \Rightarrow$

rule 8

$(q, + id, + T) \Rightarrow$

rule 4

$(q, id, T) \Rightarrow$

rule 2 second choice

$(q, id, F) \Rightarrow$

rule 3 second choice

$(q, id, id) \Rightarrow$

rule 8

$(q, \varepsilon, \varepsilon)$

normal termination when PDA stack and input string are empty (ε); otherwise, abnormal termination.

Formal PDA Definition

PDA is 7-tuple = $(K, \Sigma, H, \delta, q_0, Z_0, F)$

Where:

K = finite set of states

Σ = finite set of input tokens

H = finite push-down stack alphabet

δ = finite set of moves

q_0 = initial state

Z_0 = initial symbol on push-down stack

F = finite set of final states

Ex 2, PDA

Example 2 is a PDA that recognizes a string of 0's and 1's that is immediately followed by the same string in reverse.

Ex 2 PDA implements $\{w w^R \mid w \in \{0,1\}^*\}$

Ex 2, PDA Moves

	From			To			
Row	State	Input	Stack	State	Stac	State	Stack
1	P	0	R	P	^k BR		
2	P	1	R	P	GR		
3	P	0	B	P	BB	Q	ϵ
4	P	0	G	P	BG		
5	P	1	B	P	GB		
6	P	1	G	P	GG	Q	ϵ
7	P	ϵ	R	Q	ϵ		
8	Q	0	B	Q	ϵ		
9	Q	1	G	Q	ϵ		
10	Q	ϵ	R	Q	ϵ		

Ex 2, Recognize 001100

(P, 001100, R)	\Rightarrow	(P, 01100, BR)	By row 1
or	\Rightarrow	(Q, 001100, ε)	By row 7 (block)
(P, 01100, BR)	\Rightarrow	(P, 1100, BBR)	By row 3a
or	\Rightarrow	(Q, 1100, R)	By row 3b
(Q, 1100, R)	\Rightarrow	(Q, 1100, ε)	By row 10 (block)
(P, 1100, BBR)	\Rightarrow	(P, 100, GBBR)	By row 5
(P, 100, GBBR)	\Rightarrow	(P, 00, GGBBR)	By row 6a
or	\Rightarrow	(Q, 00, BBR)	By row 6b
(P, 00, GGBBR)	\Rightarrow	(P, 0, BGGBBR)	By row 4
(P, 0, BGGBBR)	\Rightarrow	(P, ε , BBGGBBR)	By row 3a (block)
or	\Rightarrow	(Q, ε , GGBBR)	By row 3b (block)
(Q, 00, BBR)	\Rightarrow	(Q, 0, BR)	By row 8
(Q, 00, BR)	\Rightarrow	(Q, ε , R)	By row 8
(Q, ε , R)	\Rightarrow	(Q, ε , ε)	By row 10 (accept)