Syntax Analysis
Part II

Chapter 4

Bottom-Up Parsing

- LR methods (Left-to-right, Reftmost derivation)
  - SLR, Canonical LR, LALR
- Other special cases:
  - Shift-reduce parsing
  - Operator-precedence parsing
    - Special case of shift-reduce parsing

Shift-Reduce Parsing

Grammar:

$S \rightarrow \mathbf{a}\ A\ B\ \mathbf{e}$

$A \rightarrow A\ \mathbf{b}\ \mathbf{c} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

These match production's right-hand sides

Reducing a sentence:

$\mathbf{a}\ \underline{\mathbf{b}}\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ \mathbf{e}$

$\mathbf{a}\ \underline{A\ \mathbf{b}\ \mathbf{c}}\ \mathbf{d}\ \mathbf{e}$

$\mathbf{a}\ A\ \underline{\mathbf{d}}\ \mathbf{e}$

$\underline{\mathbf{a}\ A\ B\ \mathbf{e}}$

S

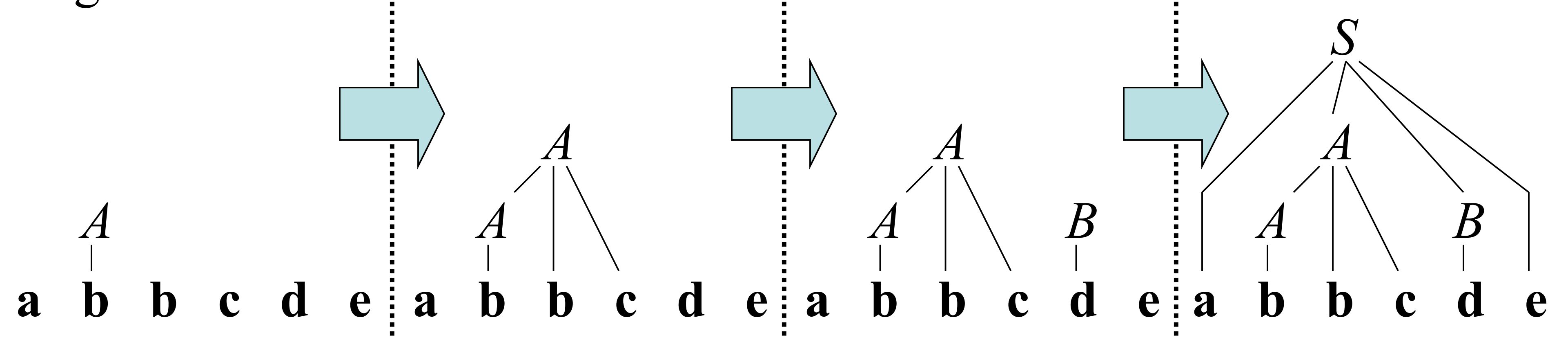
Shift-reduce corresponds to a rightmost derivation:

$S \Rightarrow_{rm} \mathbf{a}\ A\ B\ \mathbf{e}$

$\Rightarrow_{rm} \mathbf{a}\ A\ \mathbf{d}\ \mathbf{e}$

$\Rightarrow_{rm} \mathbf{a}\ A\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ \mathbf{e}$

$\Rightarrow_{rm} \mathbf{a}\ \mathbf{b}\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ \mathbf{e}$



Handles

A *handle* is a substring of grammar symbols in a *right-sentential form* that matches a right-hand side of a production

Grammar:

$S \rightarrow \mathbf{a}\ A\ B\ \mathbf{e}$

$A \rightarrow A\ \mathbf{b}\ \mathbf{c} \mid \mathbf{b}$

$B \rightarrow \mathbf{d}$

$\mathbf{a}\ \underline{\mathbf{b}}\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ \mathbf{e}$

$\mathbf{a}\ \underline{A\ \mathbf{b}\ \mathbf{c}}\ \mathbf{d}\ \mathbf{e}$

$\mathbf{a}\ A\ \underline{\mathbf{d}}\ \mathbf{e}$

$\underline{\mathbf{a}\ A\ B\ \mathbf{e}}$

S

Handle

$\mathbf{a}\ \underline{\mathbf{b}}\ \mathbf{b}\ \mathbf{c}\ \mathbf{d}\ \mathbf{e}$

$\mathbf{a}\ A\ \underline{\mathbf{b}}\ \mathbf{c}\ \mathbf{d}\ \mathbf{e}$

$\mathbf{a}\ A\ A\ \mathbf{e}$

… ?

NOT a handle, because further reductions will fail (result is not a sentential form)

Stack Implementation of Shift-Reduce Parsing

Grammar:
$E \rightarrow E + E$
$E \rightarrow E * E$
$E \rightarrow (E)$
$E \rightarrow \textbf{id}$

Stack	Input	Action
$	**id+id*id$**	shift
**$**$\underline{\textbf{id}}$	**+id*id$**	reduce $E \rightarrow \textbf{id}$
**$**$E$	**+id*id$**	shift
$**$E$+**	**id*id$**	shift
**$**$E$**+**$\underline{\textbf{id}}$	***id$**	reduce $E \rightarrow \textbf{id}$
**$**$E$**+**$E$	***id$**	shift (or reduce?)
$**$E$**+**$E$***	**id$**	shift
**$**$E$**+**$E$*****$\underline{\textbf{id}}$	**$**	reduce $E \rightarrow \textbf{id}$
**$**$E$**+**$\underline{E*E}$	**$**	reduce $E \rightarrow E * E$
**$**$\underline{E+E}$	**$**	reduce $E \rightarrow E + E$
**$**$E$	**$**	accept

Find handles to reduce

How to resolve conflicts?

Conflicts

- Shift-reduce and reduce-reduce conflicts are caused by
  - The limitations of the LR parsing method (even when the grammar is unambiguous)
  - Ambiguity of the grammar

Reduce-Reduce Conflicts

Stack	Input	Action
$	**aa$**	shift
$<u>a</u>	**a$**	reduce $A \rightarrow$ **a** <u>or</u> $B \rightarrow$ **a** ?

Grammar:

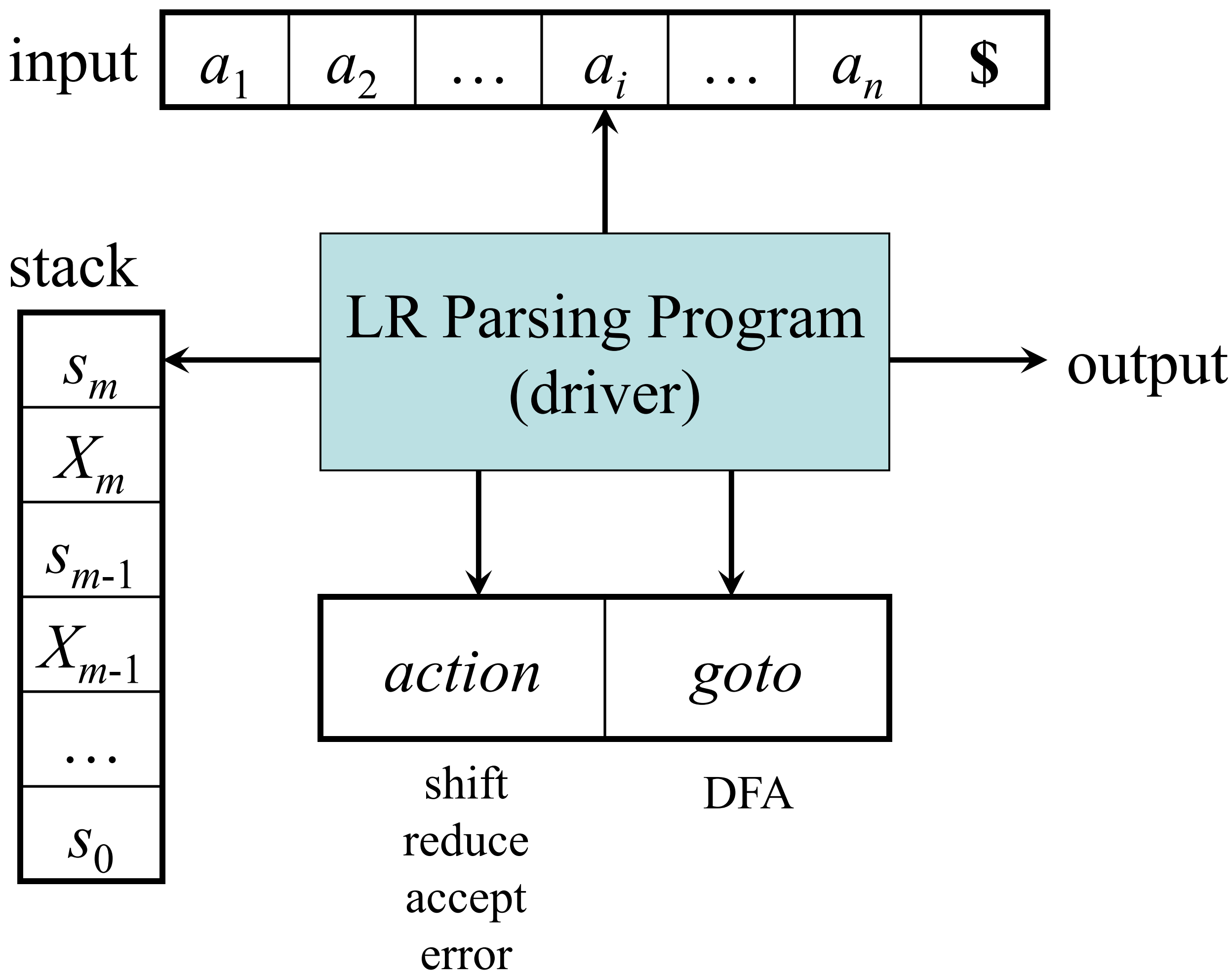
$C \rightarrow A\ B$

$A \rightarrow$ **a**

$B \rightarrow$ **a**

Resolve in favor of reduce $A \rightarrow$ **a**, otherwise we're stuck!

Model of an LR Parser



LR(0) Items of a Grammar

- An *LR(0) item* of a grammar G is a production of G with a • at some position of the right-hand side
- Thus, a production

  $$A \to X\ Y\ Z$$

  has four items:

  $$[A \to \bullet\ X\ Y\ Z]$$
  $$[A \to X \bullet Y\ Z]$$
  $$[A \to X\ Y \bullet Z]$$
  $$[A \to X\ Y\ Z \bullet]$$
- Note that production $A \to \varepsilon$ has one item $[A \to \bullet]$

The Closure Operation for LR(0) Items

1. Start with $closure(I) = I$
2. If $[A \to \alpha \bullet B\beta] \in closure(I)$ then for each production $B \to \gamma$ in the grammar, add the item $[B \to \bullet\gamma]$ to I if not already in I
3. Repeat 2 until no new items can be added

The Closure Operation (Example)

$closure(\{[E' \to \bullet E]\}) =$

$\{ [E' \to \bullet\ E] \}$ ⇒ $\{ [E' \to \bullet\ E]$
$[E \to \bullet\ E + T]$
$[E \to \bullet\ T] \}$

Add $[E \to \bullet\gamma]$

⇒ $\{ [E' \to \bullet\ E]$
$[E \to \bullet\ E + T]$
$[E \to \bullet\ T]$
$[T \to \bullet\ T * F]$
$[T \to \bullet\ F] \}$

Add $[T \to \bullet\gamma]$

⇒ $\{ [E' \to \bullet\ E]$
$[E \to \bullet\ E + T]$
$[E \to \bullet\ T]$
$[T \to \bullet\ T * F]$
$[T \to \bullet\ F]$
$[F \to \bullet\ \mathbf{(}\ E\ \mathbf{)}]$
$[F \to \bullet\ \mathbf{id}] \}$

Add $[F \to \bullet\gamma]$

Grammar:

$E \to E + T \mid T$

$T \to T * F \mid F$

$F \to \mathbf{(}\ E\ \mathbf{)}$

$F \to \mathbf{id}$

The Goto Operation for LR(0) Items

1. For each item $[A \rightarrow \alpha \bullet X \beta] \in I$, add the set of items *closure*($\{[A \rightarrow \alpha X \bullet \beta]\}$) to *goto*($I$,$X$) if not already there
2. Repeat step 1 until no more items can be added to *goto*(I,X)

The Goto Operation Example

Suppose $I = \{ [E' \to E \bullet], [E \to E \bullet + T] \}$

Then $goto(I,+) = closure(\{[E \to E + \bullet\ T]\}) = \{ [E \to E + \bullet\ T]$
$[T \to \bullet\ T * F]$
$[T \to \bullet\ F]$
$[F \to \bullet\ \mathbf{(}\ E\ \mathbf{)}]$
$[F \to \bullet\ \mathbf{id}] \}$

Grammar:

$E \to E + T \mid T$

$T \to T * F \mid F$

$F \to \mathbf{(}\ E\ \mathbf{)}$

$F \to \mathbf{id}$

Constructing the set of LR(0) Items of a Grammar

1. The grammar is augmented with a new start symbol S' and production $S' \to S$
2. Initially, set $C = closure(\{[S' \to \bullet S]\})$
3. For each set of items $I \in C$ and each grammar symbol $X \in (N \cup T)$ such that $goto(I,X) \notin C$ and $goto(I,X) \neq \emptyset$, add the set of items $goto(I,X)$ to C
4. Repeat 3 until no more sets can be added to C

Example SLR Grammar and LR(0) Items

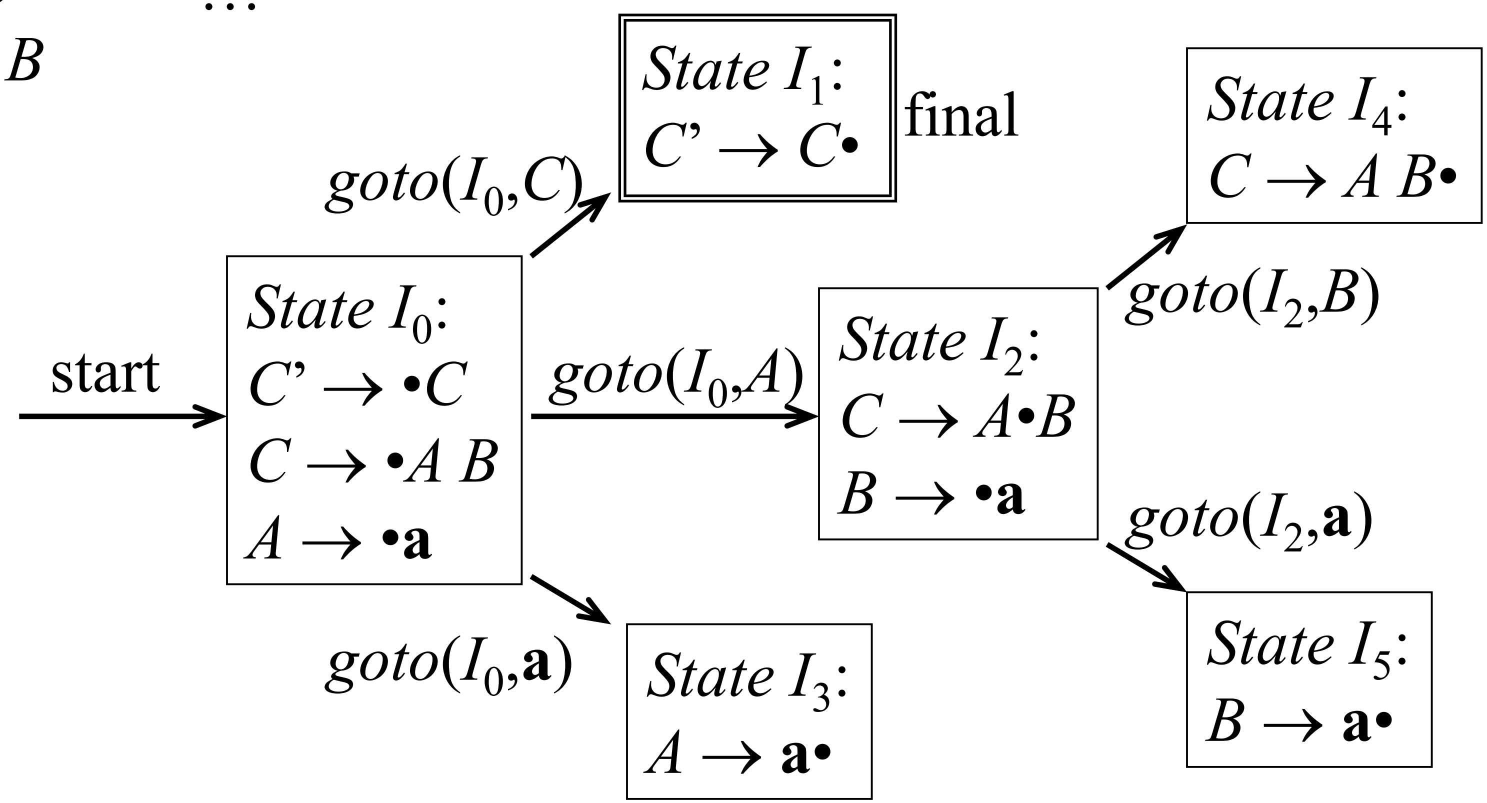
Augmented grammar:

1. $C' \to C$
2. $C \to A\ B$
3. $A \to \mathbf{a}$
4. $B \to \mathbf{a}$

$I_0 = closure(\{[C' \to \bullet C]\})$

$I_1 = \text{goto}(I_0, C) = closure(\{[C' \to C\bullet]\})$

…



Constructing SLR Parsing Tables

1. Augment the grammar with $S' \to S$
2. Construct the set $C=\{I_0,I_1,\ldots,I_n\}$ of *LR(0) items*
3. If $[A\to\alpha\bullet a\beta] \in I_i$ and $goto(I_i,a)=I_j$ then set $action[i,a]=$shift j
4. If $[A\to\alpha\bullet] \in I_i$ then set $action[i,a]=$reduce $A\to\alpha$ for all $a \in$ FOLLOW(A) (apply only if $A\neq S'$)
5. If $[S'\to S\bullet]$ is in I_i then set $action[i,\$]=$accept
6. If $goto(I_i,A)=I_j$ then set $goto[i,A]=j$
7. Repeat 3-6 until no more entries added
8. The initial state i is the I_i holding item $[S'\to\bullet S]$

Example SLR Parsing Table

State I_0:
$C' \to \bullet C$
$C \to \bullet A\ B$
$A \to \bullet\mathbf{a}$

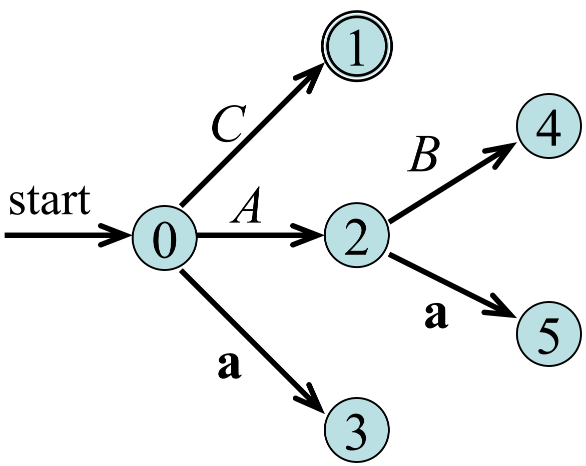
State I_1:
$C' \to C\bullet$

State I_2:
$C \to A\bullet B$
$B \to \bullet\mathbf{a}$

State I_3:
$A \to \mathbf{a}\bullet$

State I_4:
$C \to A\ B\bullet$

State I_5:
$B \to \mathbf{a}\bullet$



	a	**$**	*C*	*A*	*B*
0	s3		1	2	
1		acc			
2	s5				4
3	r3				
4		r2			
5		r4			

Grammar:
1. $C' \to C$
2. $C \to A\ B$
3. $A \to \mathbf{a}$
4. $B \to \mathbf{a}$

SLR Parsing

Configuration (= LR parser state):

$$(\underbrace{s_0\, s_1\, s_2\, \ldots\, s_m,}_{\text{stack}}\quad \underbrace{a_i\, a_{i+1}\, \ldots\, a_n\, \mathbf{\$}}_{\text{input}})$$

If $action[s_m,a_i]$ = shift s, then push s, and advance input:

$(s_0\, s_1\, s_2\, \ldots\, s_m\, s,\quad a_{i+1}\, \ldots\, a_n\, \mathbf{\$})$

If $action[s_m,a_i]$ = reduce A $\rightarrow$ β and $goto[s_{m\text{-}r},A] = s$.

If r=|β| then pop r symbols, and push s:

$(s_0\, s_1\, s_2\, \ldots\, s_{m\text{-}r}\, s,\quad a_i\, a_{i+1}\, \ldots\, a_n\, \mathbf{\$})$

If $action[s_m,a_i]$ = accept, then stop

If $action[s_m,a_i]$ = error, then attempt recovery

SLR and Ambiguity

- Every SLR grammar is unambiguous, but **not** every unambiguous grammar is SLR
- Consider for example the unambiguous grammar

$$S \to L = R \mid R$$
$$L \to * R \mid \mathbf{id}$$
$$R \to L$$

I_0:	I_1:	I_2:	I_3:	I_4:	I_5:	I_6:	I_7:	I_8:	I_9:
$S' \to \bullet S$	$S' \to S\bullet$	$S \to L\bullet{=}R$	$S \to R\bullet$	$L \to *\bullet R$	$L \to \mathbf{id}\bullet$	$S \to L{=}\bullet R$	$L \to *R\bullet$	$R \to L\bullet$	$S \to L{=}R\bullet$
$S \to \bullet L{=}R$		$R \to L\bullet$		$R \to \bullet L$		$R \to \bullet L$			
$S \to \bullet R$				$L \to \bullet *R$		$L \to \bullet *R$			
$L \to \bullet *R$				$L \to \bullet\mathbf{id}$		$L \to \bullet\mathbf{id}$			
$L \to \bullet\mathbf{id}$									
$R \to \bullet L$									

action[2,=]=s6
action[2,=]=r6

no

Has no SLR parsing table

LR(1) Grammars

- SLR too simple
- LR(1) parsing uses lookahead to avoid unnecessary conflicts in parsing table
- LR(1) item = LR(0) item + lookahead

LR(0) item:	LR(1) item:
$[A \to \alpha \bullet \beta]$	$[A \to \alpha \bullet \beta, a]$

LR(1) Items

- An *LR(1) item*
  $[A \rightarrow \alpha \bullet \beta, a]$
  contains a *lookahead* terminal a, meaning α already on top of the stack, expect to see βa
- For items of the form
  $[A \rightarrow \alpha \bullet, a]$
  the lookahead a is used to reduce $A \rightarrow \alpha$ only if the next input is a
- For items of the form
  $[A \rightarrow \alpha \bullet \beta, a]$
  with $\beta \neq \varepsilon$ the lookahead has no effect

The Closure Operation for LR(1) Items

1. Start with *closure*(I) = I
2. If $[A \rightarrow \alpha \bullet B\beta, a] \in$ *closure*(I) then for each production $B \rightarrow \gamma$ in the grammar and each terminal $b \in \mathrm{FIRST}(\beta a)$, add the item $[B \rightarrow \bullet\gamma, b]$ to I if not already in I
3. Repeat 2 until no new items can be added

The Goto Operation for LR(1) Items

1. For each item $[A\to\alpha\bullet X\beta, a] \in I$, add the set of items *closure*($\{[A\to\alpha X\bullet\beta, a]\}$) to *goto*($I$,$X$) if not already there
2. Repeat step 1 until no more items can be added to *goto*(I,X)

Constructing the set of LR(1) Items of a Grammar

1. Augment the grammar with a new start symbol S' and production $S' \rightarrow S$
2. Initially, set $C = closure(\{[S' \rightarrow \bullet S, \mathbf{\$}]\})$ (this is the start state of the DFA)
3. For each set of items $I \in C$ and each grammar symbol $X \in (N \cup T)$ such that $goto(I,X) \notin C$ and $goto(I,X) \neq \emptyset$, add the set of items $goto(I,X)$ to C
4. Repeat 3 until no more sets can be added to C

Example Grammar and LR(1) Items

- Unambiguous LR(1) grammar:

  $S \rightarrow L = R \mid R$

  $L \rightarrow * \, R \mid \textbf{id}$

  $R \rightarrow L$

- Augment with $S' \rightarrow S$
- LR(1) items (next slide)

I_0:	$[S' \to \bullet S,$	$\mathbf{\$}]$ goto$(I_0,S)=I_1$
	$[S \to \bullet L{=}R,$	$\mathbf{\$}]$ goto$(I_0,L)=I_2$
	$[S \to \bullet R,$	$\mathbf{\$}]$ goto$(I_0,R)=I_3$
	$[L \to \bullet {*}R,$	$\mathbf{=/\$}]$ goto$(I_0,*)=I_4$
	$[L \to \bullet \mathbf{id},$	$\mathbf{=/\$}]$ goto$(I_0,\mathbf{id})=I_5$
	$[R \to \bullet L,$	$\mathbf{\$}]$ goto$(I_0,L)=I_2$
I_1:	$[S' \to S\bullet,$	$\mathbf{\$}]$
I_2:	$[S \to L\bullet{=}R,$	$\mathbf{\$}]$ goto$(I_2,=)=I_6$
	$[R \to L\bullet,$	$\mathbf{\$}]$
I_3:	$[S \to R\bullet,$	$\mathbf{\$}]$
I_4:	$[L \to {*}\bullet R,$	$\mathbf{=/\$}]$ goto$(I_4,R)=I_7$
	$[R \to \bullet L,$	$\mathbf{=/\$}]$ goto$(I_4,L)=I_8$
	$[L \to \bullet {*}R,$	$\mathbf{=/\$}]$ goto$(I_4,*)=I_4$
	$[L \to \bullet \mathbf{id},$	$\mathbf{=/\$}]$ goto$(I_4,\mathbf{id})=I_5$
I_5:	$[L \to \mathbf{id}\bullet,$	$\mathbf{=/\$}]$
I_6:	$[S \to L{=}\bullet R,$	$\mathbf{\$}]$ goto$(I_6,R)=I_4$
	$[R \to \bullet L,$	$\mathbf{\$}]$ goto$(I_6,L)=I_{10}$
	$[L \to \bullet {*}R,$	$\mathbf{\$}]$ goto$(I_6,*)=I_{11}$
	$[L \to \bullet \mathbf{id},$	$\mathbf{\$}]$ goto$(I_6,\mathbf{id})=I_{12}$
I_7:	$[L \to {*}R\bullet,$	$\mathbf{=/\$}]$
I_8:	$[R \to L\bullet,$	$\mathbf{=/\$}]$
I_9:	$[S \to L{=}R\bullet,$	$\mathbf{\$}]$
I_{10}:	$[R \to L\bullet,$	$\mathbf{\$}]$
I_{11}:	$[L \to {*}\bullet R,$	$\mathbf{\$}]$ goto$(I_{11},R)=I_{13}$
	$[R \to \bullet L,$	$\mathbf{\$}]$ goto$(I_{11},L)=I_{10}$
	$[L \to \bullet {*}R,$	$\mathbf{\$}]$ goto$(I_{11},*)=I_{11}$
	$[L \to \bullet \mathbf{id},$	$\mathbf{\$}]$ goto$(I_{11},\mathbf{id})=I_{12}$
I_{12}:	$[L \to \mathbf{id}\bullet,$	$\mathbf{\$}]$
I_{13}:	$[L \to {*}R\bullet,$	$\mathbf{\$}]$

Constructing Canonical LR(1) Parsing Tables

1. Augment the grammar with $S' \rightarrow S$
2. Construct the set $C=\{I_0,I_1,\ldots,I_n\}$ of LR(1) items
3. If $[A\rightarrow\alpha\bullet a\beta,\ b] \in I_i$ and $goto(I_i,a)=I_j$ then set $action[i,a]$=shift j
4. If $[A\rightarrow\alpha\bullet,\ a] \in I_i$ then set $action[i,a]$=reduce $A\rightarrow\alpha$ (apply only if $A\neq S'$)
5. If $[S'\rightarrow S\bullet,\ \$]$ is in I_i then set $action[i,\$]$=accept
6. If $goto(I_i,A)=I_j$ then set $goto[i,A]=j$
7. Repeat 3-6 until no more entries added
8. The initial state i is the I_i holding item $[S'\rightarrow\bullet S,\$]$

Example LR(1) Parsing Table

Grammar:

1. S’ → *S*
2. *S* → *L* = *R*
3. *S* → *R*
4. *L* → * *R*
5. *L* → **id**
6. *R* → *L*

	id	*	=	$	*S*	*L*	*R*
0	s5	s4			1	2	3
1				acc			
2			s6	r6			
3				r3			
4	s5	s4				8	7
5			r5	r5			
6	s12	s11				10	4
7			r4	r4			
8			r6	r6			
9				r2			
10				r6			
11	s12	s11				10	13
12				r5			
13				r4			

LALR(1) Grammars

- LR(1) parsing tables have many states
- LALR(1) parsing (Look-Ahead LR) combines LR(1) states to reduce table size
- Less powerful than LR(1)
  - Will not introduce shift-reduce conflicts, because shifts do not use lookaheads
  - May introduce reduce-reduce conflicts, but seldom do so for grammars of programming languages

Constructing LALR(1) Parsing Tables

1. Construct sets of LR(1) items
2. Combine LR(1) sets with sets of items that share the same first part

$$
\begin{array}{lll}
I_4: & [L \to *\bullet R, & =] \\
 & [R \to \bullet L, & =] \\
 & [L \to \bullet *R, & =] \\
 & [L \to \bullet \mathbf{id}, & =] \\
 & & \\
I_{11}: & [L \to *\bullet R, & \$] \\
 & [R \to \bullet L, & \$] \\
 & [L \to \bullet *R, & \$] \\
 & [L \to \bullet \mathbf{id}, & \$]
\end{array}
$$

$$
\begin{array}{ll}
[L \to *\bullet R, & =/\$] \\
[R \to \bullet L, & =/\$] \\
[L \to \bullet *R, & =/\$] \\
[L \to \bullet \mathbf{id}, & =/\$]
\end{array}
$$

Shorthand for two items in the same set

Example LALR(1) Grammar

- Unambiguous LR(1) grammar:

  $S \to L = R \mid R$

  $L \to * R \mid \mathbf{id}$

  $R \to L$

- Augment with $S' \to S$
- LALR(1) items (next slide)

I_0: $[S' \to \bullet S,$ $\mathbf{\$}]$ goto$(I_0,S)=I_1$
$[S \to \bullet L\mathbf{=}R,$ $\mathbf{\$}]$ goto$(I_0,L)=I_2$
$[S \to \bullet R,$ $\mathbf{\$}]$ goto$(I_0,R)=I_3$
$[L \to \bullet\mathbf{*}R,$ $\mathbf{=}]$ goto$(I_0,\mathbf{*})=I_4$
$[L \to \bullet\mathbf{id},$ $\mathbf{=}]$ goto$(I_0,\mathbf{id})=I_5$
$[R \to \bullet L,$ $\mathbf{\$}]$ goto$(I_0,L)=I_2$

I_1: $[S' \to S\bullet,$ $\mathbf{\$}]$

I_2: $[S \to L\bullet\mathbf{=}R,$ $\mathbf{\$}]$ goto$(I_0,\mathbf{=})=I_6$
$[R \to L\bullet,$ $\mathbf{\$}]$

I_3: $[S \to R\bullet,$ $\mathbf{\$}]$

I_4: $[L \to \mathbf{*}\bullet R,$ $\mathbf{=}/\mathbf{\$}]$ goto$(I_4,R)=I_7$
$[R \to \bullet L,$ $\mathbf{=}/\mathbf{\$}]$ goto$(I_4,L)=I_9$
$[L \to \bullet\mathbf{*}R,$ $\mathbf{=}/\mathbf{\$}]$ goto$(I_4,\mathbf{*})=I_4$
$[L \to \bullet\mathbf{id},$ $\mathbf{=}/\mathbf{\$}]$ goto$(I_4,\mathbf{id})=I_5$

I_5: $[L \to \mathbf{id}\bullet,$ $\mathbf{=}/\mathbf{\$}]$

I_6: $[S \to L\mathbf{=}\bullet R,$ $\mathbf{\$}]$ goto$(I_6,R)=I_8$
$[R \to \bullet L,$ $\mathbf{\$}]$ goto$(I_6,L)=I_9$
$[L \to \bullet\mathbf{*}R,$ $\mathbf{\$}]$ goto$(I_6,\mathbf{*})=I_4$
$[L \to \bullet\mathbf{id},$ $\mathbf{\$}]$ goto$(I_6,\mathbf{id})=I_5$

I_7: $[L \to \mathbf{*}R\bullet,$ $\mathbf{=}/\mathbf{\$}]$

I_8: $[S \to L\mathbf{=}R\bullet,$ $\mathbf{\$}]$

I_9: $[R \to L\bullet,$ $\mathbf{=}/\mathbf{\$}]$

Shorthand for two items

$[R \to L\bullet,$ $\mathbf{=}]$
$[R \to L\bullet,$ $\mathbf{\$}]$

Example LALR(1) Parsing Table

Grammar:
1. $S' \rightarrow S$
2. $S \rightarrow L = R$
3. $S \rightarrow R$
4. $L \rightarrow$ ***** R
5. $L \rightarrow$ **id**
6. $R \rightarrow L$

	id	*****	**=**	**$**	*S*	*L*	*R*
0	s5	s4			1	2	3
1				acc			
2			s6	r6			
3				r3			
4	s5	s4				9	7
5			r5	r5			
6	s5	s4				9	8
7			r4	r4			
8				r2			
9			r6	r6			

LL, SLR, LR, LALR Summary

- LL parse tables computed using FIRST/FOLLOW
  - Nonterminals × terminals → productions
  - Computed using FIRST/FOLLOW
- LR parsing tables computed using closure/goto
  - LR states × terminals → shift/reduce actions
  - LR states × terminals → goto state transitions
- A grammar is
  - LL(1) if its LL(1) parse table has no conflicts
  - SLR if its SLR parse table has no conflicts
  - LALR(1) if its LALR(1) parse table has no conflicts
  - LR(1) if its LR(1) parse table has no conflicts

LL, SLR, LR, LALR Grammars