

COSE312: Compilers

Lecture 6 — LR Parsing with Ambiguous Grammars

Hakjoo Oh
2023 Spring

Parsing with Ambiguous Grammars

In programming languages, ambiguous grammars provide more natural and concise specification:

$$E \rightarrow E + E \mid E * E \mid (E) \mid \text{id} \quad \text{vs.} \quad \begin{array}{l} E \rightarrow E + T \mid T \\ T \rightarrow T * F \mid F \\ F \rightarrow (E) \mid \text{id} \end{array}$$

Conflicts

$$I_0 : \begin{array}{l} E' \rightarrow .E \\ E \rightarrow .E + E \\ E \rightarrow .E * E \\ E \rightarrow .(E) \\ E \rightarrow .id \end{array}$$

$$I_1 : \begin{array}{l} E' \rightarrow E. \\ E \rightarrow E. + E \\ E \rightarrow E. * E \end{array}$$

$$I_2 : \begin{array}{l} E \rightarrow (.E) \\ E \rightarrow .E + E \\ E \rightarrow .E * E \\ E \rightarrow .(E) \\ E \rightarrow .id \end{array}$$

$$I_3 : \boxed{E \rightarrow id.}$$

$$I_4 : \begin{array}{l} E \rightarrow E + .E \\ E \rightarrow .E + E \\ E \rightarrow .E * E \\ E \rightarrow .(E) \\ E \rightarrow .id \end{array}$$

$$I_5 : \begin{array}{l} E \rightarrow E * .E \\ E \rightarrow .E + E \\ E \rightarrow .E * E \\ E \rightarrow .(E) \\ E \rightarrow .id \end{array}$$

$$I_6 : \begin{array}{l} E \rightarrow (E.) \\ E \rightarrow E. + E \\ E \rightarrow E. * E \end{array}$$

$$I_7 : \begin{array}{l} E \rightarrow E + E. \\ E \rightarrow E. + E \\ E \rightarrow E. * E \end{array}$$

$$I_8 : \begin{array}{l} E \rightarrow E * E. \\ E \rightarrow E. + E \\ E \rightarrow E. * E \end{array}$$

$$I_9 : \boxed{E \rightarrow (E).}$$

SLR Parsing Table

STATE	id	+	*	()	\$	<i>E</i>
0	s3			s2			g1
1		s4	s5			acc	
2	s3			s2			g6
3		r4	r4		r4	r4	
4	s3			s2			g7
5	s3			s2			g8
6		s4	s5		s9		
7		s4, r1	s5, r1		r1	r1	
8		s4, r2	s5, r2		r2	r2	
9		r3	r3		r3	r3	

Resolving Conflicts with Precedence and Associativity

Conflicts are resolved by assuming that

- $*$ takes precedence over $+$, and
- $+$ and $*$ are left-associative.

Resolving Conflicts with Precedence

The parsing process has shift/reduce conflicts for input $\text{id} + \text{id} * \text{id}$:

Stack	Symbols	Input	Action
0		$\text{id} + \text{id} * \text{id}\$$	shift to 3
0 3	id	$+ \text{id} * \text{id}\$$	reduce by 4
0 1	E	$+ \text{id} * \text{id}\$$	shift to 4
0 1 4	$E +$	$\text{id} * \text{id}\$$	shift to 3
0 1 4 3	$E + \text{id}$	$* \text{id}\$$	reduce by 4
0 1 4 7	$E + E$	$* \text{id}\$$	shift to 5, reduce by 1

Which is the correct action?

Resolving Conflicts with Precedence

When we choose the shift action:

Stack	Symbols	Input	Action
0		id + id * id\$	shift to 3
0 3	id	+id * id\$	reduce by 4
0 1	<i>E</i>	+id * id\$	shift to 4
0 1 4	<i>E</i> +	id * id\$	shift to 3
0 1 4 3	<i>E</i> + id	*id\$	reduce by 4
0 1 4 7	<i>E</i> + <i>E</i>	*id\$	<u>shift to 5</u> , reduce by 1
0 1 4 7 5	<i>E</i> + <i>E</i> *	id\$	shift to 3
0 1 4 7 5 3	<i>E</i> + <i>E</i> * id	\$	reduce by 4
0 1 4 7 5 8	<i>E</i> + <i>E</i> * <i>E</i>	\$	reduce by 2
0 1 4 7	<i>E</i> + <i>E</i>	\$	reduce by 1
0 1	<i>E</i>	\$	accept

Resolving Conflicts with Precedence

When we choose the reduce action:

Stack	Symbols	Input	Action
0		id + id * id\$	shift to 3
0 3	id	+id * id\$	reduce by 4
0 1	<i>E</i>	+id * id\$	shift to 4
0 1 4	<i>E</i> +	id * id\$	shift to 3
0 1 4 3	<i>E</i> + id	*id\$	reduce by 4
0 1 4 7	<i>E</i> + <i>E</i>	*id\$	shift to 5, <u>reduce by 1</u>
0 1	<i>E</i>	*id\$	shift to 5
0 1 5	<i>E</i> *	id\$	shift to 3
0 1 5 3	<i>E</i> * id	\$	reduce by 4
0 1 5 8	<i>E</i> * <i>E</i>	\$	reduce by 2
0 1	<i>E</i>	\$	accept

Resolving Conflicts with Precedence

Take the shift action when the parser is at state 7 and the next input symbol is *:

STATE	id	+	*	()	\$	<i>E</i>
0	<i>s3</i>			<i>s2</i>			<i>g1</i>
1		<i>s4</i>	<i>s5</i>			<i>acc</i>	
2	<i>s3</i>			<i>s2</i>			<i>g6</i>
3		<i>r4</i>	<i>r4</i>		<i>r4</i>	<i>r4</i>	
4	<i>s3</i>			<i>s2</i>			<i>g7</i>
5	<i>s3</i>			<i>s2</i>			<i>g8</i>
6		<i>s4</i>	<i>s5</i>		<i>s9</i>		
7		<i>s4, r1</i>	<i>s5</i>		<i>r1</i>	<i>r1</i>	
8		<i>s4, r2</i>	<i>s5, r2</i>		<i>r2</i>	<i>r2</i>	
9		<i>r3</i>	<i>r3</i>		<i>r3</i>	<i>r3</i>	

Resolving Conflicts with Associativity

The parsing goes into a shift/reduce conflict for input **id + id + id**:

Stack	Symbols	Input	Action
0 1 4 7	<i>E</i> + <i>E</i>	+id\$	shift to 4, reduce by 1

Which is the correct action?

Resolving Conflicts with Associativity

STATE	id	+	*	()	\$	<i>E</i>
0	s3			s2			g1
1		s4	s5			acc	
2	s3			s2			g6
3		r4	r4		r4	r4	
4	s3			s2			g7
5	s3			s2			g8
6		s4	s5		s9		
7		r1	s5		r1	r1	
8		r2	r2		r2	r2	
9		r3	r3		r3	r3	

The “Dangling-Else” Ambiguity

stmt \rightarrow if *expr* then *stmt*
 | if *expr* then *stmt* else *stmt*
 | other

The “Dangling-Else” Ambiguity

$$\begin{array}{lcl} stmt & \rightarrow & \text{if } expr \text{ then } stmt \\ & | & \text{if } expr \text{ then } stmt \text{ else } stmt \\ & | & \text{other} \end{array}$$

Simplified grammar:

$$\begin{array}{lcl} S' & \rightarrow & S \\ S & \rightarrow & i S e S \mid i S \mid a \end{array}$$

Conflicts

LR(0) states include the state:

$$I_4 = \begin{array}{l} S \rightarrow iS.eS \\ S \rightarrow iS. \end{array}$$

Conflicts

LR(0) states include the state:

$$I_4 = \begin{array}{l} S \rightarrow iS.eS \\ S \rightarrow iS. \end{array}$$

The conflict in the SLR parsing table:

STATE	<i>i</i>	<i>e</i>	<i>a</i>	\$	<i>S</i>
4		s5, r2		r2	

Which is the correct action?

Summary

- Ambiguous grammar is useful for programming languages.
- We can use the ambiguous grammar in LR parsing by specifying precedence and associativity rules.