COSE312: Compilers

Lecture 6 — Syntax Analysis (4): Ambiguous Grammars

Hakjoo Oh
2025 Spring

Parsing with Ambiguous Grammars

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

$$\boldsymbol{E} \rightarrow \boldsymbol{E} + \boldsymbol{E} \mid \boldsymbol{E} * \boldsymbol{E} \mid (\boldsymbol{E}) \mid \mathbf{id} \tag{1}$$

$$\begin{array}{lcl} \boldsymbol{E} & \rightarrow & \boldsymbol{E} + \boldsymbol{T} \mid \boldsymbol{T} \\ \boldsymbol{T} & \rightarrow & \boldsymbol{T} * \boldsymbol{F} \mid \boldsymbol{F} \\ \boldsymbol{F} & \rightarrow & (\boldsymbol{E}) \mid \mathbf{id} \end{array} \tag{2}$$

- The ambiguous grammar in (1) does not specify the associativity or precedence of the operators $+$ and $*$.
- The unambiguous grammar in (2) gives $+$ lower precedence than $*$, makes both operators left associative.
- In practice, we prefer to use the ambiguous grammar because we can often enforce the associativity and precedence w/o changing grammar.

Conflicts

The augmented grammar:

$$(0)\ E' \to E, \quad E \to (1)\ E + E \mid (2)\ E * E \mid (3)\ (E) \mid (4)\ \mathbf{id}$$

The sets of LR(0) items:

I_0 :
$E' \to .E$
$E \to .E + E$
$E \to .E * E$
$E \to .(E)$
$E \to .\mathbf{id}$

I_1 :
$E' \to E.$
$E \to E. + E$
$E \to E. * E$

I_2 :
$E \to (.E)$
$E \to .E + E$
$E \to .E * E$
$E \to .(E)$
$E \to .\mathbf{id}$

I_3 :
$E \to \mathbf{id}.$

I_4 :
$E \to E + .E$
$E \to .E + E$
$E \to .E * E$
$E \to .(E)$
$E \to .\mathbf{id}$

I_5 :
$E \to E * .E$
$E \to .E + E$
$E \to .E * E$
$E \to .(E)$
$E \to .\mathbf{id}$

I_6 :
$E \to (E.)$
$E \to E. + E$
$E \to E. * E$

I_7 :
$E \to E + E.$
$E \to E. + E$
$E \to E. * E$

I_8 :
$E \to E * E.$
$E \to E. + E$
$E \to E. * E$

I_9 :
$E \to (E).$

Which states cause conflicts during the SLR parsing?

SLR Parsing Table

STATE	id	+	*	(	)	$	E
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 $\mathbf{id} + \mathbf{id} * \mathbf{id}$:

Stack	Symbols	Input	Action
0		$\mathbf{id} + \mathbf{id} * \mathbf{id\$}$	shift to 3
0 3	$\mathbf{id}$	$+\mathbf{id} * \mathbf{id\$}$	reduce by 4
0 1	$\boldsymbol{E}$	$+\mathbf{id} * \mathbf{id\$}$	shift to 4
0 1 4	$\boldsymbol{E}+$	$\mathbf{id} * \mathbf{id\$}$	shift to 3
0 1 4 3	$\boldsymbol{E} + \mathbf{id}$	$*\mathbf{id\$}$	reduce by 4
0 1 4 7	$\boldsymbol{E} + \boldsymbol{E}$	$*\mathbf{id\$}$	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		$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	shift to 3
0 3	$\mathbf{id}$	$+\mathbf{id} * \mathbf{id}\$$	reduce by 4
0 1	$\boldsymbol{E}$	$+\mathbf{id} * \mathbf{id}\$$	shift to 4
0 1 4	$\boldsymbol{E}+$	$\mathbf{id} * \mathbf{id}\$$	shift to 3
0 1 4 3	$\boldsymbol{E} + \mathbf{id}$	$*\mathbf{id}\$$	reduce by 4
0 1 4 7	$\boldsymbol{E} + \boldsymbol{E}$	$*\mathbf{id}\$$	shift to 5, reduce by 1
0 1 4 7 5	$\boldsymbol{E} + \boldsymbol{E}*$	$\mathbf{id}\$$	shift to 3
0 1 4 7 5 3	$\boldsymbol{E} + \boldsymbol{E} * \mathbf{id}$	$\$$	reduce by 4
0 1 4 7 5 8	$\boldsymbol{E} + \boldsymbol{E} * \boldsymbol{E}$	$\$$	reduce by 2
0 1 4 7	$\boldsymbol{E} + \boldsymbol{E}$	$\$$	reduce by 1
0 1	$\boldsymbol{E}$	$\$$	accept

Resolving Conflicts with Precedence

When we choose the reduce action:

Stack	Symbols	Input	Action
0		$\mathbf{id} + \mathbf{id} * \mathbf{id}\$$	shift to 3
0 3	$\mathbf{id}$	$+\mathbf{id} * \mathbf{id}\$$	reduce by 4
0 1	$\boldsymbol{E}$	$+\mathbf{id} * \mathbf{id}\$$	shift to 4
0 1 4	$\boldsymbol{E}+$	$\mathbf{id} * \mathbf{id}\$$	shift to 3
0 1 4 3	$\boldsymbol{E} + \mathbf{id}$	$*\mathbf{id}\$$	reduce by 4
0 1 4 7	$\boldsymbol{E} + \boldsymbol{E}$	$*\mathbf{id}\$$	shift to 5, reduce by 1
0 1	$\boldsymbol{E}$	$*\mathbf{id}\$$	shift to 5
0 1 5	$\boldsymbol{E}*$	$\mathbf{id}\$$	shift to 3
0 1 5 3	$\boldsymbol{E} * \mathbf{id}$	$\$$	reduce by 4
0 1 5 8	$\boldsymbol{E} * \boldsymbol{E}$	$\$$	reduce by 2
0 1	$\boldsymbol{E}$	$\$$	accept

Resolving Conflicts with Precedence

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

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

Resolving Conflicts with Associativity

The parsing goes into a shift/reduce conflict for input $\mathbf{id} + \mathbf{id} + \mathbf{id}$:

Stack	Symbols	Input	Action
0 1 4 7	$\boldsymbol{E} + \boldsymbol{E}$	$+\mathbf{id\$}$	shift to 4, reduce by 1

Which is the correct action?

Resolving Conflicts with Associativity

Take the reduce action when the parser is at state 7 and the next input symbol is $+$:

STATE	id	$+$	$*$	(	)	$	E
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		$s4, r2$	$s5, r2$		$r2$	$r2$	
9		$r3$	$r3$		$r3$	$r3$	

Exercise

Suppose the parse is at state **8**.

- Which is correct when the next input is $+$? Explain with an example.
- Which is correct when the next input is $*$? Explain with an example.

The "Dangling-Else" Ambiguity

Grammar for conditional statements:

$$
\begin{array}{rcl}
\boldsymbol{stmt} & \rightarrow & \textbf{if } \boldsymbol{expr} \textbf{ then } \boldsymbol{stmt} \\
& | & \textbf{if } \boldsymbol{expr} \textbf{ then } \boldsymbol{stmt} \textbf{ else } \boldsymbol{stmt} \\
& | & \textbf{other}
\end{array}
$$

Consider the statement:

$$\textbf{if } \boldsymbol{E_1} \textbf{ then if } \boldsymbol{E_2} \textbf{ then } \boldsymbol{S_1} \textbf{ else } \boldsymbol{S_2}$$

which has two parse trees.

The "Dangling-Else" Ambiguity

Grammar (simplified and augmented):

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

LR(0) states:

$$I_0 = \boxed{\begin{array}{l} S' \rightarrow .S \\ S \rightarrow .iSeS \\ S \rightarrow .iS \\ S \rightarrow .a \end{array}} \quad I_1 = \boxed{S' \rightarrow S.} \quad I_2 = \boxed{\begin{array}{l} S \rightarrow i.SeS \\ S \rightarrow i.S \\ S \rightarrow .iSeS \\ S \rightarrow .iS \\ S \rightarrow .a \end{array}} \quad I_3 = \boxed{S \rightarrow a.}$$

$$I_4 = \boxed{\begin{array}{l} S \rightarrow iS.eS \\ S \rightarrow iS. \end{array}} \quad I_5 = \boxed{\begin{array}{l} S \rightarrow iSe.S \\ S \rightarrow .iSeS \\ S \rightarrow .iS \\ S \rightarrow .a \end{array}} \quad I_6 = \boxed{S \rightarrow iSeS.}$$

Which states generate conflicts?

Exercise

- $\boldsymbol{FOLLOW(S) =}$
- Complete the SLR parsing table:

	Action				Goto
STATE	i	e	a	$	S
0	$s2$		$s3$		1
1				acc	
2	$s2$		$s3$		4
3		$r3$		$r3$	
4					
5	$s2$		$s3$		6
6		$r1$		$r1$	

- Which action is correct when conflicts occur? Remove the ambiguity in the parsing table.

Exercise

Describe the parsing actions on input $iiaea$:

Stack	Symbols	Input	Action
0		$iiaea\$$	shift

Exercise

The ambiguity of the grammar

$$
\begin{array}{rcl}
\textit{stmt} & \rightarrow & \textbf{if } \textit{expr} \textbf{ then } \textit{stmt} \\
& | & \textbf{if } \textit{expr} \textbf{ then } \textit{stmt} \textbf{ else } \textit{stmt} \\
& | & \textbf{other}
\end{array}
$$

can be eliminated by introducing auxiliary nonterminals M (*matched statement*) and U (*unmatched statement*):

$$
\begin{array}{rcl}
S & \rightarrow & M \\
S & \rightarrow & U \\
M & \rightarrow & \textbf{if } \textit{expr} \textbf{ then ? else ?} \\
M & \rightarrow & \textbf{other} \\
U & \rightarrow & \textbf{if } \textit{expr} \textbf{ then ?} \\
U & \rightarrow & \textbf{if } \textit{expr} \textbf{ then ? else ?}
\end{array}
$$

Summary

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