EC3204: Programming Languages and Compilers

Lecture 8 — Syntax Analysis (4): *LR Parsing with Ambiguous Grammars*

Sunbeom So
Fall 2024

LR Parsing with Ambiguous Grammars

- LR parsing techniques are popular because they are more power powerful than LL methods.
- (Challenge) To apply LR techniques, ambiguity must be resolved. Ambiguous expression grammar:

$$E \to E + E \mid E * E \mid (E) \mid \mathbf{id}$$

  Unambiguous grammar:

$$\begin{array}{rcl} E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & (E) \mid \mathbf{id} \end{array}$$

- Designing an equivalent unambiguous grammar is hard and sometimes impossible.

LR techniques can be extended to ambiguous grammars with **disambiguating rules**.

Conflicts due to the Ambiguity

The sets of LR(0) items for the ambiguous expression grammar:

$$I_0: \boxed{\begin{array}{l} E' \to .E \\ E \to .E + E \\ E \to .E * E \\ E \to .(E) \\ E \to .\mathbf{id} \end{array}} \quad I_1: \boxed{\begin{array}{l} E' \to E. \\ E \to E. + E \\ E \to E. * E \end{array}} \quad I_2: \boxed{\begin{array}{l} E \to (.E) \\ E \to .E + E \\ E \to .E * E \\ E \to .(E) \\ E \to .\mathbf{id} \end{array}}$$

$$I_3: \boxed{E \to \mathbf{id}.} \quad I_4: \boxed{\begin{array}{l} E \to E + .E \\ E \to .E + E \\ E \to .E * E \\ E \to .(E) \\ E \to .\mathbf{id} \end{array}} \quad I_5: \boxed{\begin{array}{l} E \to E * .E \\ E \to .E + E \\ E \to .E * E \\ E \to .(E) \\ E \to .\mathbf{id} \end{array}}$$

$$I_6: \boxed{\begin{array}{l} E \to (E.) \\ E \to E. + E \\ E \to E. * E \end{array}} \quad I_7: \boxed{\begin{array}{l} E \to E + E. \\ E \to E. + E \\ E \to E. * E \end{array}} \quad I_8: \boxed{\begin{array}{l} E \to E * E. \\ E \to E. + E \\ E \to E. * E \end{array}} \quad I_9: \boxed{E \to (E).}$$

Q. Which states have shift/reduce conflicts in SLR parsing?

Conflicts due to the Ambiguity

The SLR parsing table for the ambiguous grammar:

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

- Since $+, * \in FOLLOW(E)$, we have $r1$ at $(7, +), (7, *)$.
- Since $+, * \in FOLLOW(E)$, we have $r2$ at $(8, +), (8, *)$.
- Q. How to resolve the conflicts without modifying the grammar?
- A. The conflicts can be resolved by assuming the two rules:
  1. $*$ takes precedence over $+$, and
  2. $+$ and $*$ are left-associative.

Resolving Conflicts with Precedence

The shift/reduce conflict will occur for the 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	E	$+\mathbf{id} * \mathbf{id}\$$	shift to 4
0 1 4	$E+$	$\mathbf{id} * \mathbf{id}\$$	shift to 3
0 1 4 3	$E + \mathbf{id}$	$*\mathbf{id}\$$	reduce by 4
0 1 4 7	$E + E$	$*\mathbf{id}\$$	shift to 5, reduce by 1

Which one is the correct action?
What happens if we choose the shift (or reduce)?

Resolving Conflicts with Precedence

If we choose the shift action, * will have the precedence.

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

If we choose the reduce action, $+$ will have the precedence.

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

In conclusion, the parsing action at the entry $(\mathbf{7}, *)$ must be $\boldsymbol{s5}$, in order to give precedence $*$ over $+$.

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		$s4, r1$	$s5, \cancel{r1}$		$r1$	$r1$	
8		$s4, r2$	$s5, r2$		$r2$	$r2$	
9		$r3$	$r3$		$r3$	$r3$	

Resolving Conflicts with Associativity

The parsing goes into a shift/reduce conflict for the 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 4, reduce by 1

Which one is the correct action?

What happens if we choose the shift (or reduce)?

Resolving Conflicts with Associativity

If we choose the shift action, $+$ will be right-associative.

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 4, reduce by 1
0 1 4 7 4	$\boldsymbol{E} + \boldsymbol{E}+$	$\mathbf{id}\$$	shift to 3
0 1 4 7 4 3	$\boldsymbol{E} + \boldsymbol{E} + \mathbf{id}$	$\$$	reduce by 4
0 1 4 7 4 7	$\boldsymbol{E} + \boldsymbol{E} + \boldsymbol{E}$	$\$$	reduce by 1
0 1 4 7	$\boldsymbol{E} + \boldsymbol{E}$	$\$$	reduce by 1
0 1	$\boldsymbol{E}$	$\$$	accept

Resolving Conflicts with Associativity

If we choose the reduce action, $+$ will be left-associative.

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 4, reduce by 1
0 1	$\boldsymbol{E}$	$+\mathbf{id}\$$	shift to 4
0 1 4	$\boldsymbol{E}+$	$\mathbf{id}\$$	shift to 3
0 1 4 3	$\boldsymbol{E} + \mathbf{id}$	$\$$	reduce by 4
0 1 4 7	$\boldsymbol{E} + \boldsymbol{E}$	$\$$	reduce by 1
0 1	$\boldsymbol{E}$	$\$$	accept

Resolving Conflicts with Associativity

In conclusion, the parsing action at the entry $(\mathbf{7}, +)$ must be $\boldsymbol{r1}$, in order to make $+$ left-associative.

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

Resolving Conflicts in the "Dangling-Else" Grammar

Consider the grammar for specifying statements:

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

where **other** stands for any other statements (e.g., assignments).

- Q. Is this grammar ambiguous?

Resolving Conflicts in the “Dangling-Else” Grammar

Consider the grammar for specifying statements:

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

where **other** stands for any other statements (e.g., assignments).

- Q. Is this grammar ambiguous?
- A. Ambiguous. Consider $\underline{\textbf{if } E_1 \textbf{ then if } E_2 \textbf{ then } S_1 \textbf{ else } S_2}$.
  - 1st interpretation: $\textbf{if } E_1 \textbf{ then } (\textbf{if } E_2 \textbf{ then } S_1 \textbf{ else } S_2)$
  - 2nd interpretation: $\textbf{if } E_1 \textbf{ then } (\textbf{if } E_2 \textbf{ then } S_1) \textbf{ else } S_2$
- The first interpretation (parse tree) is more widely acceptable.
  - General rule: “Match each **else** with the closest unmatched **then**”
- The problem, where **else** clause becomes ambiguous in nested conditional statements, is called the **dangling-else** ambiguity.

Resolving Conflicts in the "Dangling-Else" Grammar

- We can resolve the "dangling-else" ambiguity by designing the equivalent unambiguous grammar.

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

Idea: the interior statemetns between **then** and **else** must not end with an "open" (unmatched) **then** branch.

Resolving Conflicts in the "Dangling-Else" Grammar

- We can resolve the "dangling-else" ambiguity by designing the equivalent unambiguous grammar.

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

Idea: the interior statemetns between **then** and **else** must not end with an "open" (unmatched) **then** branch.

- Simpler solution: specify additional disambiguating rules! To explain this point, consider the simplified grammar:

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

where i stands for **if** $expr$ **then**, S for $stmt$, e for **else**, and a for **other**, respectively.

Resolving Conflicts in the "Dangling-Else" Grammar

LR(0) states for the simplified dangling-else grammar:

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

$$I_3: \boxed{S \rightarrow a.} \qquad I_4: \boxed{\begin{array}{l} S \rightarrow iS.eS \\ S \rightarrow iS. \end{array}} \qquad I_5: \boxed{\begin{array}{l} S \rightarrow iSe.S \\ S \rightarrow .iSeS \\ S \rightarrow .iS \\ S \rightarrow .a \end{array}} \qquad I_6: \boxed{S \rightarrow iSeS.}$$

Q. Which states have shift/reduce conflicts in SLR parsing?

Resolving Conflicts in the “Dangling-Else” Grammar

SLR parsing table for the simplified dangling-else grammar:

STATE	i	e	a	$\$$	S
$\mathbf{0}$	$s2$		$s3$		$g1$
$\mathbf{1}$				acc	
$\mathbf{2}$	$s2$		$s3$		$g4$
$\mathbf{3}$		$r3$		$r3$	
$\mathbf{4}$		$s5, r2$		$r2$	
$\mathbf{5}$	$s2$		$s2$		$g6$
$\mathbf{6}$		$r1$		$r1$	

- At position $(4, e)$, we have $r2$ since $FOLLOW(S) = \{e, \$\}$. As a result, there exist shift/reduce conflicts at $(4, e)$.
- Q. Which one is the correct action? How to resolve the conflict without modifying the grammar?
- A. Shift. We typically prefer shift over reduce unless imposed by other rules – see p.26 at https://www-labs.iro.umontreal.ca/~monnier/6232/ocamlyacc-tutorial.pdf

Summary

- LR parsing techniques are useful, but dealing with ambiguous grammar is hard.
- We can extend LR techniques using disambiguating rules (e.g., precedence and associativity rules, preferring shift).