Error Recovery Strategies and LR Parsing Conflicts

Lecture 10

Section 4.1.4, 4.4.5, 4.5.4

Rokan Uddin Faruqui

Associate Professor
Dept of Computer Science and Engineering
University of Chittaong, Bangladesh
Email: *rokan@cu.ac.bd*

Outline

Error Recovery

Once an error is detected during parsing, how should the parser recover?

Error Recovery

- Parser to quit with an informative error message when it detects the first error.
- Additional errors are often uncovered
  - if the parser can restore itself to a state where processing of the input can continue
  - may provide meaningful diagnostic information.
- If errors pile up, it is better for the compiler to give up

Error Recovery Strategies

Panic Mode
- Parser discards input symbols one at a time until one of a designated set of synchronizing tokens is found.
- Synchronizing tokens are usually delimiters, such as ; or }, whose role in the source program is clear and unambiguous.

Phrase-Level Recovery
- Parser may perform local correction on the remaining input
- May replace a prefix of the remaining input by some string that allows the parser to continue
- A typical local correction is to replace a comma by a semicolon, delete an extraneous semicolon, or insert a missing semicolon.

Error Recovery Strategies

E—>E+T

Error Productions
- By anticipating common error, grammar is augmented with productions that generate the erroneous constructs
- Parser can then generate appropriate error diagnostics about the erroneous construct that has been recognized in the input

Global Correction Given an incorrect input string x and grammar G, these algorithms will and a parse tree for a related string y, such that the number of *insertions*, *deletions*, and *changes* of tokens required to transform x into y is as small as possible.

Outline

Error Recovery in Predictive Parsing

An error is detected during predictive parsing when the terminal on top of the stack does not match the next input symbol or when nonterminal A is on top of the stack, a is the next input symbol, and $M[A;a]$ is error (i.e., the parsing-table entry is empty).

Error Recovery Strategies in Predictive Parsing

Panic Mode

1. Place all symbols in $FOLLOW(A)$ into the synchronizing set for nonterminal A. Skip tokens until an element of $FOLLOW(A)$ is seen and pop A from the stack.
2. Additional symbols such as delimiters in synchronizing set.
3. Add symbols in $FIRST(A)$ to the synchronizing set for nonterminal A.
4. If nonterminal is **nullable**, then ϵ production can be uses as a default.
5. If a terminal on top of the stack cannot be matched, a simple idea is to pop the terminal, issue a message saying that the terminal was inserted, and continue parsing

Example

Example

- Let the grammar be

$$
\begin{aligned}
E &\to T\ E' \\
E' &\to +\ T\ E' \mid \varepsilon \\
T &\to F\ T' \\
T' &\to *\ F\ T' \mid \varepsilon \\
F &\to (\ E\ ) \mid \mathbf{id}
\end{aligned}
$$

Example

Example (Parse Table)

- Recall

Nonterminal	Nullable	FIRST	FOLLOW
E	No	$\{(, \textbf{id}\}$	$\{\$,)\}$
E'	Yes	$\{+\}$	$\{\$,)\}$
T	No	$\{(, \textbf{id}\}$	$\{\$,), +\}$
T'	Yes	$\{*\}$	$\{\$,), +\}$
F	No	$\{(, \textbf{id}\}$	$\{*, \$,), +\}$

Example

Example (Parse Table)

	+	*	(	)	**id**	\$
E			$E \rightarrow T\ E'$		$E \rightarrow T\ E'$	
E'	$E' \rightarrow + T\ E'$			$E' \rightarrow \varepsilon$		$E' \rightarrow \varepsilon$
T			$T \rightarrow F\ T'$		$T \rightarrow F\ T'$	
T'	$T' \rightarrow \varepsilon$	$T' \rightarrow * F\ T'$		$T' \rightarrow \varepsilon$		$T' \rightarrow \varepsilon$
F			$F \rightarrow (E)$		$F \rightarrow \mathbf{id}$	

Example

Example (Parse Table with Error Recovery)

	+	*	(	)	**id**	$
E			$E \to T\ E'$	sync	$E \to T\ E'$	sync
E'	$E' \to + T\ E'$			$E' \to \varepsilon$		$E' \to \varepsilon$
T	sync		$T \to F\ T'$	sync	$T \to F\ T'$	sync
T'	$T' \to \varepsilon$	$T' \to * F\ T'$		$T' \to \varepsilon$		$T' \to \varepsilon$
F	sync	sync	$F \to (E)$	sync	$F \to \textbf{id}$	sync

STACK	INPUT	REMARK
$E\,\$$	$)\,\mathbf{id} * + \mathbf{id}\,\$$	error, skip)
$E\,\$$	$\mathbf{id} * + \mathbf{id}\,\$$	**id** is in FIRST(E)
$TE'\,\$$	$\mathbf{id} * + \mathbf{id}\,\$$	
$FT'E'\,\$$	$\mathbf{id} * + \mathbf{id}\,\$$	
$\mathbf{id}\ T'E'\$$	$\mathbf{id} * + \mathbf{id}\,\$$	
$T'E'\,\$$	$* + \mathbf{id}\,\$$	
$* FT'E'\,\$$	$* + \mathbf{id}\,\$$	
$FT'E'\,\$$	$+ \mathbf{id}\,\$$	error, $M[F, +] =$ synch
$T'E'\,\$$	$+ \mathbf{id}\,\$$	F has been popped
$E'\,\$$	$+ \mathbf{id}\,\$$	
$+ TE'\,\$$	$+ \mathbf{id}\,\$$	
$TE'\,\$$	$\mathbf{id}\,\$$	
$FT'E'\,\$$	$\mathbf{id}\,\$$	
$\mathbf{id}\,T'E'\,\$$	$\mathbf{id}\,\$$	
$T'E'\,\$$	$\$$	
$E'\,\$$	$\$$	
$\$$	$\$$	

Outline

Example

Example (A Simplified Grammar)

- We may simplify our grammar to

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

- In this form, the precedence rules for + and * are not implicit.
- They must be incorporated into the tables.

Shift/Reduce Conflicts

- It is possible that a cell will contain both a shift operation and a reduce operation.
- This is called a shift/reduce conflict.
- To choose between “shift” and “reduce,” each case must be considered on its own merit.
- Consider the case of $E \rightarrow E + E \mid E * E$ and the inputs **a + b ∗ c** and **a ∗ b + c**.

4+5*3 => 27, 19

Reduce/Reduce Conflicts

- It is possible that a cell will contain two different reduce operations.
- This is called a reduce/reduce conflict.
- This occurs when a sequence of tokens matches the right-hand sides of two different productions at the same time.
- For each such conflict in the table, we must choose which reduction to apply.
- The presence of a reduce/reduce conflict indicates that you should find another grammar.

Reduce/Reduce Conflicts

- It is possible that a cell will contain two different reduce operations.
- This is called a reduce/reduce conflict.
- This occurs when a sequence of tokens matches the right-hand sides of two different productions at the same time.
- For each such conflict in the table, we must choose which reduction to apply.
- The presence of a reduce/reduce conflict indicates that you should find another grammar.

Example

Example (The Action and Goto Tables)

State	Action						Goto
	+	*	(	)	**id**	$	E
0			s2		s3		1
1	s4	s5				acc	
2			s2		s3		6
3	r4	r4		r4		r4	
4			s2		s3		7
5			s2		s3		8
6	s4	s5		s9			
7	s4/r1	s5/r1		r1		r1	
8	s4/r2	s5/r2		r2		r2	
9	r3	r3		r3		r3	

Outline

Example

Example (Shift/Reduce Conflicts and Associativity)

- The shift/reduce conflict in cell $(7, +)$ is between shifting a $+$ and reducing by

$$E \to E + E.$$

- If we choose "shift," then we will make addition right associative.
- If we choose "reduce," then we will make addition left associative.
- The case is similar in cell $(8, *)$ regarding multiplication.

Example

Example (Shift/Reduce Conflicts and Precedence)

- The shift/reduce conflict in cell $(8, +)$ is between shifting a $+$ and reducing by
$$E \to E * E.$$
4+5*5

- If we choose "shift," then we will give multiplication a higher precedence than addition.
- If we choose "reduce," then we will give addition a higher precedence than multiplication.
- The case is similar in cell $(7, *)$.

Example

Example (The Action and Goto Tables)

State	Action						Goto
	+	*	(	)	**id**	$	E
0			s2		s3		1
1	s4	s6				acc	
2			s2		s3		6
3	r4	r4		r4		r4	
4			s2		s3		7
5			s2		s3		8
6	s4	s5		s9			
7	r1	s5		r1		r1	
8	r2	r2		r2		r2	
9	r3	r3		r3		r3	

Outline

Assignment

Assignment

- Let the grammar be

$$S \rightarrow (\ L\ )\ |\ \mathbf{id}$$
$$L \rightarrow L\ ,\ S\ |\ S$$

- Parse the expression $\mathbf{(id,id)}$.

(continued...)

Assignment

Assignment

- The grammar for if and if-else statements is

$$
\begin{aligned}
S &\rightarrow \texttt{if (} E \texttt{) } S \\
&\mid \texttt{if (} E \texttt{) } S \texttt{ else } S \\
&\mid \textbf{id = num} \\
E &\rightarrow \textbf{id == num}
\end{aligned}
$$

  where S is a statement and E is an expression (boolean).

- Write the action and goto tables for this grammar.
- Find the shift/reduce conflict(s) and decide how to handle them.