LANGUAGE PROCESSORS

UNIT 8: ERROR HANDLING

David Griol Barres
dgriol@inf.uc3m.es
Computer Science Department
Carlos III University of Madrid
Leganés (Spain)

OUTLINE

- Introduction
- Recovering from errors
- Error Recovery Strategies
  - Panic mode
  - Phrase level
  - Error Productions
  - Global correction
  - Top-down predictive parser: error detection
  - Error detection and recovery in LL parsers
  - Error detection and recovery in LR parsers

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Introduction

- **A parser should**:
  - Find errors as soon as possible.
  - Report errors with a comprehensive message.
  - Try to parse as much of the code as possible in order to find as many errors as possible.
  - Avoid cascading errors.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Recovering from errors

- For many minor errors, the parser can "fix" the program by guessing at what was intended and reporting a warning, but allowing compilation to proceed.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error Recovery Strategies

- No universally acceptable strategy.
- **Common strategies**:
  1. Panic mode.
  2. Phrase level.
  3. Error productions.
  4. Global correction.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Panic mode

- **Characteristics**:
  - The simplest method to implement.
  - Can be used by most parsing methods.
  - It does not go into an infinite loop.
  - An adequate method in situations where multiple errors in the same statement are rare.

- On discovering an error, the parsers discards input symbols one at a time until a synchronizing token (e. g., delimiters).

- **Drawbacks**:
  - A considerable amount of input is skipped without checking it for additional errors.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Phrase level

- **Characteristics**:
  - It can correct any input string.
- On discovering an error, a parser may perform local correction on the remaining input to allow the parser to continue (e. g., replace a comma by a semicolon).
- **Drawbacks**:
  - The difficulty in coping with situations where the actual error occurred before the point of detection.
  - Some replacements may lead to infinite loops.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error Productions

If we know what errors are common in a language, we can augment the grammar with productions that generate the erroneous constructs in order to detect the error.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Global correction

- Use algorithms for choosing the minimal sequence of changes to obtain a globally least-cost correction.
- Drawbacks:
  - Too costly to implement in terms of time and space.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Top-down predictive parser: error detection

An error is detected 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 the pasing table entry M[A,a] is empty.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error detection in LL parsers

- Grammar:

```
S::=a A S | b A
A::=c A | d
```

- Table:

Σ_N	a	b	c	d
S	aAS	bA		
A			cA	d

- Input: a b ...
- State of the parser when the error is detected.

```
Stack Input
$S          a b ...
$SAa a b ...
$SA  b ...
```

- **Error:** There is a b in the input instead of a c or d.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error recovery in LL parsers

- Panic-mode heuristics:
  - For a nonterminal A, we could place all the symbols in Follow(A) into its synchronizing set.
  - We could also use the symbols in First(A) as a synchronizing set for re-starting the parse of A.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error recovery in LL parsers

- Phrase-level recovery:
  - Fill in the blank entries in the parsing table with pointers to error routines:
    - The routines may change, insert, or delete symbols on the input and issue error messages.
    - They may also pop from the stack.
  - Protect against loops!
    - Any recovery action eventually results on an input symbol being consumed or the stack being shortened.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error detection in LR parsers

- Grammar

S::=A ·A
A::=x ·A | y

- Input: xy
- State of the parser when the error is detected:

Stack	Input
0x3A6	$
0A2	$
0A2	$

	action			goto	
	x	y	$	S	A
0	d3	d4		1	2
1			acpt		
2	d3	d4	err3	5	
3	d3	d4		6	6
4	r3	r3	r3		7
5	r1				8
6	r2	r2	r2		

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error handling in LR parsing

- Error detection:
  - An error entry in the parsing action table.
  - A canonical LR parsing will never make a reduction before announcing an error.
- Panic-mode error recovery
  - Scan down the stack until a state s with a goto on a particular nonterminal A is found.
  - Discard zero or more input symbols until a symbol a is found that can follow A.
  - Push goto[s,A] onto the stack and continue the parsing.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es

Error handling in LR parsing

- Phrase-level recovery
  - Appropriate recovery procedure: the top of the stack and/or first input symbols would be modified in an appropriate way for each error entry.
  - Any reduction called for by an LR parser is surely correct.
  - Recovery actions may include insertion or deletion of symbols from the stack or the input or both, or alteration and transposition of input symbols.
  - Popping a stack state that covers a nonterminal should be avoided because it eliminates a construct that has already been successfully parsed.

David Griol Barres Carlos III University of Madrid dgriol@inf.uc3m.es