Syntactic Error Recovery

66.648 Compiler Design Lecture (02/12/97)

Computer Science

Rensselaer Polytechnic

Lecture Outline

- Error Recovery Methods
- Classification of context free languages
- Administration

Syntax Errors

Three levels of handling syntax errors:

1) Error Detection: Detect and report errors as early as possible in the edit-compile cycle.

2) Error Recovery: Report the first error to the user and throw him into an editor. (or) Recover from the first error by ignoring additional tokens.

3) Error Correction: Correct the user's input program (by making an educated guess and continue) - this happens in a shell level.

Error Recovery in Bottom-up Parsing

These parsing techniques have the correct prefix property I.e., when an error is detected at certain point, the prefix of that program is a legal program.

Two commonly used error recovery techniques:

a) Panic Mode

b) Error Productions

Panic Mode Error Recovery

Basic Idea: After an error has occurred, pop zero or more symbols from the stack and discard zero or more tokens from the input buffer until we reach a point where parsing can continue.

1. Define a set of panic nonterminals among the given set of nonterminals.
2. On error, pop parser stack till we reach a state S expecting a goto transition on one or more panic nonterminals. Parsing is stopped if no such state is found.

Panic --Contd

3. Discard tokens from input buffer till we find a synchronizing token I.e. a token that belongs to FOLLOW(A), for a panic nonterminal found in step 2.
4. Push A and the state GOTO(S,A) on the parse stack and resume parsing.

Example

S --> E

E--> E + T | T   T is the panic nonterminal

T --> T* num | num

STACK	Input Buffer	ACTION
$	num * * num	shift
$num	** num	reduce
$T	** num	shift
$T *	* num	error

Example -Contd

Now pop stack till we get T and discard input till we get the follow(T)

$T	* num	shift
$T *	num	shift
$T * num		reduce
$T		reduce
$E		

Error Recovery - via Error Productions

1. Add error productions of the form A--> alpha error beta, where error is taken as a special nonterminal symbol.
2. On error pop parse stack till we reach a state S that can perform a goto transition on the error terminal. Parsing is terminated if no such state is found.
3. Perform the shift on error token. S'=goto(S,error)
4. Discard tokens from input buffer till we find a

Error - Contd

Token for which a legal parse action can be performed from state S', and resume normal parsing.

Example:

S--> E

E--> E + T | T

T --> T * num | T error num | num

Example-Contd

STACK	Input Buffer	ACTION
$	num * * num	shift
$num	* * num	reduce
$T	* * num	shift
$T *	* num	error
$T error	* num	discard input tokens
$T error	num	shift

YACC Examples

Comments and Feedback

- Project 2 is out.
- We have finished chapter 4.Please keep reading chapter 4 and understand the material. Work out as many exercises as you can.
- We move onto chapter 5 next.