Error recovery and repair

**Recovery:** Goal is to continue a correct parse as soon as possible. *Why?*

**Repair:** Pretending the input was correct (by deleting or inserting tokens).
- Two goals: Better recovery, and more useful error messages.

Recovery may discard correctly parsed input (on the stack); repair attempts not to.
What information do we have?

• In LL parsing: Current prediction says what *could* be next.

• In LR parsing: We also have legal next tokens.

An “insert-correctable” language allows recovery by inserting (never deleting) tokens.
Immediate error detection

LL(1) and LALR(1) parsers never consume a token that cannot be part of a legal prefix.

However, both continue to operate while looking at an illegal lookahead!
Error recovery in recursive descent compilers

For each parsing procedure

**ValidSet:** All legal next tokens.

**FollowSet:** What can come next. We might recover by skipping forward to a member of FollowSet.

**HeaderSet:** Things not to skip ("begin", "if", . . . );

A "panic mode" recovery; repair is difficult because the parsing stack is implicit in the recursion.
LL(1) Error recovery

Like recursive descent, but the parsing stack is explicit.

- If top of stack is terminal symbol, pop
  (equivalent to inserting a token)

- If top of stack is a non-terminal,
  pop if parsing can resume
  pop if lookahead is a header symbol
  otherwise skip lookahead
LL(1) insert-only repair

*Details in "Crafting a Compiler," Ch. 17*

Each terminal has a "cost".

Compute "least cost repairs" (prefixes to insert) from non-terminals to terminals.

Search parse stack to find non-terminal with least cost of insertions to reach current lookahead.
LR error recovery

Nearly all "panic-mode" (including Yacc/Bison/Cup)

Skip to "safe" tokens (similar to "header" tokens in LL(1) recovery).

May first pop states until a goto/shift combination can consume an input.
Yacc/bison/cup LR error recovery

Special terminal "error" marks recovery points.

Pop states until "error" can be shifted.

Shift error and skip tokens until a shift is possible.
Yacc catch-all rule from vtwm window manager

twmrc : stmts
    ;

stmts : /* Empty */
    | stmts stmt
    ;

stmt : error
    | noarg
    | sarg
    | narg
and so on
CUP example from manual

stmt ::= expr SEMI | while_stmt SEMI | if_stmt SEMI | ... | error SEMI
;

U. Oregon CIS 461/561, Winter 2005
“Safe” tokens in yacc/bison/cup

Identify good places to restart parsing.

Example: semicolon (end of statement) in C, Java, ...

```
stmt: 
    NULLSTMT SEP
    | RETURN rexpression SEP
    | assignment SEP
    | ...
    | error SEP
{ yyerror("Malformed statement discarded"); } ;
```

LR error repair with continuations

"Continuation" is essentially "FollowSet", and can be calculated from CFSM. Must be chosen to avoid infinite insertions (see Crafting a Compiler, pg. 714).

Insert zero or more tokens, then delete zero or more tokens, then continue.

Worst case is inserting a whole continuation, then deleting the rest of the program!