Error repair

Good programming languages are designed with a relatively large “distance” between syntactically correct programs, to increase the likelihood that conceptual mistakes are caught as syntactic errors.

Error repair usually occurs at two levels:

Local: repairs mistakes with little global import, such as missing semicolons and undeclared variables.

Scope: repairs the program text so that scopes are correct. Errors of this kind include unbalanced parentheses and begin/end blocks.

Repair actions can be divided into insertions and deletions. Typically the compiler will use some lookahead and backtracking in attempting to make progress in the parse. There is great variation among compilers, though some languages (PL/C) carry a tradition of good error repair. Goals of error repair include:

1. No input should cause the compiler to collapse.
2. Illegal constructs are flagged.
3. Frequently occurring errors are repaired gracefully.
4. Minimal stuttering or cascading of errors.

LL-style parsing lends itself well to error repair, since the compiler uses the grammar’s rules to predict what should occur next in the input.
Augmenting recursive descent parsers for error recovery

Recursive and LL parsers are often called predictive, because they operate by predicting the next step in a derivation.

Suppose the parser is operating in procedure \( A \) for some nonterminal \( A \). If an error occurs, it seems reasonable to recover by skipping to a symbol that could follow \( A \), and then return.

\[
\begin{align*}
E & \rightarrow T \ E' \\
E' & \rightarrow + T \ E' \\
E' & \rightarrow - T \ E' \\
E' & \rightarrow \lambda \\
T & \rightarrow F \ T' \\
T' & \rightarrow \ast F \ T' \\
T' & \rightarrow / F \ T' \\
F & \rightarrow ( E ) \\
F & \rightarrow \ast \\
F & \rightarrow (, a) \\
F & \rightarrow (, a, \ast, /, +, -, $) \\
\end{align*}
\]

Procedure \( E'(\text{StopSet}) \)

- \( \text{if (LookAhead}(+)\text{)} \) then
  - call \( \text{Expect}(+) \)
  - call \( T(\{+, -\} \cup \text{StopSet}) \)
  - call \( E'(\text{StopSet}) \)
- \( \text{else} \)
  - \( \text{if (LookAhead}($, ')\text{)} \) then
    - return \( () \)
  - \( \text{else} \)
    - call \( \text{ErrorRecover}(	ext{StopSet}) \)
  - fi
fi
end

<table>
<thead>
<tr>
<th>First</th>
<th>Follow</th>
</tr>
</thead>
<tbody>
<tr>
<td>( E )</td>
<td>{ (, a) }</td>
</tr>
<tr>
<td>( E' )</td>
<td>{ +, - }</td>
</tr>
<tr>
<td>( T )</td>
<td>{ (, a) }</td>
</tr>
<tr>
<td>( T' )</td>
<td>{ *, / }</td>
</tr>
<tr>
<td>( F )</td>
<td>{ (, a) }</td>
</tr>
</tbody>
</table>