SYNTAX ANALYSIS & ERROR RECOVERY

By: Sarthak Swaroop
The main features of syntax analysis, which is done by parser, are as follows:

- Checks the Grammar
- Parse Tree Production
- Output Errors
Types of Error

There are mainly four types of error. They are as follows:

- **Lexical Error**: Such as misspelling an identifier, keyword or operator.
- **Syntactic Error**: Such as an arithmetic expression with unbalanced parentheses.
- **Semantic Error**: Such as operator applied to an incompatible operand.
- **Logical Error**: Such as infinitely recursive call.
ERROR HANDLING TECHNIQUES
1. Panic Mode

In case of an error like:

\[
a = b + c \quad // \text{no semi-colon}
\]
\[
d = e + f ;
\]

The compiler will discard all subsequent tokens till a semi-colon is encountered.

- This is a crude method but often turns out to be the best method.
- This method often skips a considerable amount of input without checking it for additional errors, it has an advantage of simplicity.

- In situations where multiple errors in the same statements are rare, this method may be quite adequate.
2. Phrase Level Recovery

- On discovering an error, a parser may perform local correction on the remaining input; that is, it may replace a prefix of the remaining input by some string that allows the parser to continue.

- For example, in case of an error like the one above, it will report the error, generate the “;”, and continue.
3. Error Production

If we have an idea of common errors that might occur, we can include the errors in the grammar at hand. For example if we have a production rule like:

$$E \rightarrow +E \mid -E \mid *E \mid /E$$

Then,  

$$a = +b;$$  
$$a = -b;$$  
$$a = *b;$$  
$$a = /b;$$
Here, the last two are error situations.

Now, we change the grammar as:

\[
E \rightarrow +E \mid -E \mid *A \mid /A \\
A \rightarrow E
\]

Hence, once it encounters \( *A \), it sends an error message asking the user if he is sure he wants to use a unary “\*”.
4. Global Correction

- We would like compiler to make as few changes as possible in processing an incorrect input string.

- There are algorithms for choosing a minimal amount of changes to obtain a globally least-cost correction. Suppose we have a line like this:

  **THIS IS A OCMPERIL SCALS.**
To correct this, there is an attractor, which checks how different tokens are from the initial inputs, checks the closest attractor to the incorrect token is.

This is more of a probabilistic type of error correction. Unfortunately, these methods are in general too costly to implement in terms of space and time.