Lecture 07 (Lexical Analysis High level Algorithm)

1 Lexical Specifications

Different token classes can be represented using different regular languages

2 Lexical Analysis Algorithm

1. Write a regular expression for each token class
2. Construct a giant union regular expression $R = R_1 + R_2 \cdots + R_n$
3. For input $x_1 x_2 \ldots x_n$, check if $x_1 \ldots x_i \in R_j$
4. If success, emit token and remove $x_1 \ldots x_j$ and repeat step 3

3 Resolving Ambiguities

1. Maximal munch - always take the longer token
2. Prioritize token classes - keyword > identifier
3. Introduce error token class - to allow for partial lexing

4 Why Use RegEx for Lexing and not CFG?

1. >> is both a right shift and closing operator for 2 template placeholders (> + >)
2. Can postpone this decision from lexing to parsing
3. Reduces complexity of lexer
4. CFG is used anyway in parser, so can keep lexer simple
5. Additionally, restricting lexing to RegEx doesn't hamper coding style of user of language, so why complicate it