CS 460 - Compilers

Arian Izadi

Spring 2024

1 Languages

Syntax is the rules for what a syntactically correct program looks like. Semantics is the meaning of a program.

When does it matter the order of evaluation (right to left vs left to right)? When the code has side effects, an example of this is postfix vs prefix increment (a++ vs ++a).

Compilers for a language L, move from front end $\rightarrow$ intermediate representation $\rightarrow$ back end.

- Front end: FIND IN BOOK
- Intermediate: FIND IN BOOK
- Back end: FIND IN BOOK

1.1 Lexical Analysis & Scanning

Lexical analysis, a scanner, is the process of converting a stream of characters into a stream of tokens.

1. Find all terminals in the grammar.
2. Write the Scanner.
   (a) Do we use a DFA, NFA, or PDA?
   (b) Look at token types. All tokens can be expressed by a regular expression.
      i. Symbols: Semicolon, commas, etc.
      ii. Keywords: for, while, etc.
      iii. Variables: x, y, etc.
      iv. Numbers: 1, 3.14, 0x64, etc.

Chomsky Language Hierarchy

- Type 0: Unrestricted (Turing Machines)
- Type 1: Context Sensitive
- Type 2: Context Free (PDA)
- Type 3: Regular Expressions (NFA, DFA)

Both RE and CFG have 1 non-terminal on the left of any combination of terminals and non-terminals on the right.

Example 1:

$S \rightarrow X \quad X \rightarrow aXb|d$ not regular: $a^n d b^n$

Example 2:

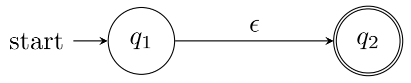
$S \rightarrow X \quad X \rightarrow aX|b$ regular: $a^* b$

Example 3:

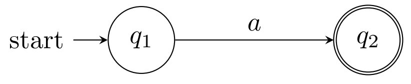
$S \rightarrow X \quad X \rightarrow aY|\epsilon \quad Y \rightarrow bX$ regular: $(ab)^*$

An NFA for recognizing tokens, construct NFA for each construct of RE.

$\mathcal{E}$:



$a \epsilon \Sigma$:



Any RE can be turned into an NFA using these rules. If all the tokens of a language are represented by RE's, $r_1, \ldots, r_n$. Create an NFA for each RE.