Chapter 2

Daniel Lee

May 7th, 2022

Introduction

- A compiler scans an input of characters and outputs a stream of words labelled by syntatic category
- A microsyntax is used to group words that have meaning within the source language
- Some words such as keywords have special meaning, which makes them reserved
- An example of this would be the *while* and *static* keywords in the Java programming language
- To recognize keywords, the scanner can either use dictionary lookup or encode keywords directly into microsyntax
- The simple lexical structure of programming languages lends itself to efficent scanners

Recognizing Words

- When we are parsing words we can view the parsing process as a series of if-else statements or a state machine
- Transition diagrams often provide a simple means of formalizing the abstractions a compiler may need to implement them
- S is the finite set of states in the recognizer, alongside with error state s_e
- Σ is the finite alphabet recognized by the recognizer
- $\delta(s, c)$ is the transition function, it maps the value of state s and c,into some state
- In state s_i with transition character c, the state makes the following transition $s_i \rightarrow_c \delta(s_i, c)$
- $s_0 \in S$ refers to initial state
- $S_a(S_a \subseteq S)$, is the set of accepting states

Example:

$S = \{s_0, s_1, s_2, s_3, ..., s_10, s_e\}$
$\Sigma = \{e, h, i, l, n, o, t, w\}$
$\delta =$
$\{ s_0 \rightarrow_n s_1, s_0 \rightarrow_w s_6, s_1 \rightarrow_e s_2, s_1 \rightarrow_o s_4, s_2 \rightarrow_w s_3$
$s_4 \rightarrow_t s_5, s_6 \rightarrow_h s_7, s_7 \rightarrow_i s_8, s_8 \rightarrow_l s_9, s_9 \rightarrow_e s_{10}$
$s_0 = s_0$
$S_A = \{s_3, s_5, s_{10}\}$

More complex words:

- For more complex words we can have the state machine accept multiple inputs
- We can vastly simplify state machines by using cycles

Practice Problems:

- Problem 1: A six-character identifier consisting of alphanumeric characters followed by zero to five-alpha numeric characters
  - $S = \{s_0, s_1, s_e\}$
  - $\Sigma = a =$ **set of all-alphabet**, $b =$ **set of all alphanumeric**
  - $s_0 = s_0$
  - $\delta = \{ s_0 \rightarrow_a s_1, s_1 \rightarrow_b s_1$
  - $S_A = s_1$
- Problem 2:
  - $S = \{s_0, s_1, s_2 s_e\}$
  - $\Sigma = (,)$
  - $s_0 = s_0$
  - $S_A = \{s_2\}$
  - $\delta = \{ s_0 \rightarrow_(s_1, s_1 \rightarrow_) s_2, s_2 \rightarrow_(s_1\}$
- Problem 3: A Pascal comment which consists of {, zero or more characters from the alphabet, and closed by }:
  - $S = \{s_0, s_1, s_2\}$
  - $\Sigma = \{\}, \{, a...z, A...Z, 0...9\}$
  - $s_0 = s_0$
  - $S_A = \{S_3\}$
  - $\delta = \{ s_0 \rightarrow_{\{} s_1$
$s_1 \rightarrow_{(a...z, A...Z, 0...9)} s_1$
$s_1 \rightarrow_{\{} s_2$

Regular Expression

- The set of all words accepted by a finite automaton, F, forms a language L(F)
- For any FA, we can describe describe the language using regular expression or RE
- The language consists of single world "new" can be described as RE, new

- A language consisting of two words, new or while can be represented as RE new|while
- new or not can be represent by RE, n(ew|ot)
- Let us consider the example of punctuation marks, a REs for punctuation may appear such as: : ; ? = > ( ) []
- Keywords may have an expression such as this: if while this integer instanceof
- more complex RE: 0 |(0|1|2|3|4|5|6|7|8|9)(0|1|2|3|4|5|6|7|8|9)$*$
- **The following operator is called a kleen operator and indicates there can be zero or more instances of a RE**