How to Describe Tokens?

- Regular languages are the most popular for specifying tokens
  - Simple and and useful theory
  - Easy to understand
  - Efficient Implementation

What is a language?

- An alphabet is a well defined set of characters. The character $\sum$ is typically used to represent an alphabet.
- A string : a finite sequence of alphabet symbols, can be ε, the empty string (Some texts use λ as the empty string)
- A language, **L**, over $\sum$ is simply any set of strings (infinite or finite) drawn from $\sum$.

What is a language? (cont'd)

Examples:

Alphabet: A-Z	Language: English
Alphabet: ASCII	Language: C++

Notation

- Languages are sets of strings (finite sequences of characters)
- Need some notation for specifying which sets we want.
- For lexical analysis, we care about regular languages.
- Regular languages can be described using regular expressions.

Regular Languages

- Each regular expression is a notation for a regular language (a set of words).
- If **A** is a regular expression, we write **L(A)** to refer to language denoted by A.
- Formally describe tokens in the language
  - Regular Expressions
  - NFA
  - DFA

Regular Expressions

- A Regular Expression is a set of rules , techniques for constructing sequences of Symbols (Strings) From an Alphabet.

If A is a regular expression, then L(A) is the language defined by that regular expression.

L(“c”) is the language with the single word “c”.
L(“i” “f”) is the language with just “if” in it.

Regular Expressions

- A Regular Expression is defined Inductively
  - a — ordinary character from ∑
  - ε — the empty string

Regular Expressions

- **R|S** = either **R** or **S**
- **RS** = **R** followed by **S** (concatenation)
- **R*** = Concatenation of **R**
  zero or more times
  (R*= ε | R|RR|RRR…)

RE Notational Shorthand

- R+ = R(R*) one or more strings of R
- R? = (R|ε) (zero or one R)
- (R) = R (grouping)
- [abcd] = (a|b|c|d) any of listed characters
- [a-z] = (a|b|c|d...|z) one character from this range
- [^ab] = anything but none of the listed chars
- [^a-z] = any character not from this range

Regular Expression

- Regular Expression, R
  - a
  - ab
  - a|b
  - (ab)*
  - (a| ε)b
  - digit = [0-9]
  - posint = digit+

- Strings in L(R)
  - “a”
  - “ab”
  - “a”, “b”
  - “”, “ab”, “abab”, ...
  - “ab”, “b”
  - “0”, “1”, “2”, ...
  - “8”, “412”, ...

Examples

- Next we will define integers in a language:
- digit = “0” | “1” | “2” | “3” | “4” | “5” | “6” | “7” | “8” | “9”
- integer = {digit}+

- Note that we can abbreviate ranges using the dash (“-”). Thus, digit = 0-9
- Relation = ‘<’ | ‘<=’ | ‘>’ | ‘>=’ | ‘<>’ | ‘=’
- Floating point numbers are not much more complicated:
- float = {digit}+ “.” {digit}+

Example: Identifiers

- Identifier: a string or letters or digits starting with a letter a non-empty string of digits

- C Identifier: [a-zA-Z_][a-zA-Z0-9_]*
-
- Digit: = ‘0’,’1’,’2’ - ‘9’

Real-world example

What is the regular expression that defines all phone numbers?

$\sum$ = { 0-9 }
Area = {digit}^{3}
Exchange = {digit}^{3}
Local = {digit}^{4}

Phone_number = “(” {Area} “)” {Exchange} {Local}

How to use REs

- We need mechanism to determine

  If an input string **w** belongs to **L(R),** the language denoted by regular expression **R**.

- C Identifier: [a-zA-Z_][a-zA-Z0-9_]*
-
- Digit: = ‘0’,’1’,’2’ - ‘9’

Finite Automata (FA)

- **Specification:** Regular Expressions
- **Implementation:** Finite Automata

Finite Automata (FA)

Finite Automation consists of

- A set of input alphabet
- A set of states
- A initial state
- A set of transitions
- A set of accepting (final) states

Finite Automata (FA)

- A finite automation accepts a string if we can follow transitions labelled with characters in the string from the start state to some accepting state.

Regular Expression Operation

- There are three basic operations in regular expression :
  - Alternation **(union)** $RE_1 \mid RE_2$
  - Concatenation **(concatenation)** $RE_1\ RE_2$
  - Repetition **(closure)** RE* (zero or more RE's)

Regular Expression Operation

If P and Q are regular expressions over Σ, then so are:

- **P | Q (union)**

  If P denotes the set {*a*,…,*e*}, Q denotes the set {*0*,…,*9*} then P + Q denotes the set {*a*,…,*e*,*0*,…,*9*}

- **PQ (concatenation)**

  If P denotes the set {*a*,…,*e*}, Q denotes the set {*0*,…,*9*} then PQ denotes the set {*a0*,…,*e0*,*a1*,…,*e9*}

- **Q* (closure)**

  If Q denotes the set {*0*,…,*9*} then Q* denotes the set {*0*,…,*9,00,…99,…*}

Examples

If $\Sigma = \{a,b\}$

- (a | b)*b
- b(a1b)*

Defining Our Language

The first thing we can define in our language are keywords. These are easy:

if | else | while | find | …

When we scan a file, we can either have a single token represent all keywords, or else break them down into groups, such as “commands”, “types”, etc.

Language Def (cont'd)

Identifiers are strings of letters, underscores, or digits beginning with a non-digit.

Letter = a-z | A-Z
digit = 0-9
Identifier = ({letter} | "_")({letter} | "_" | {digit})*

THANKS