Regular Expressions

Logistics

- Homework
  - Homework #1 due today.
  - Homework #2
    - Exercise 3.1.1 (a,b,c) – pg 89
    - Exercise 3.1.4 (a,b,c) – pg 90
    - Exercise 3.2.2 (a – d) – pg 106
    - Exercise 3.2.4 (a,b,c) – pg 106
    - Take the NFA-ε in any part of 3.2.4 and convert to a DFA.

Questions

- Any questions before we start?

Languages

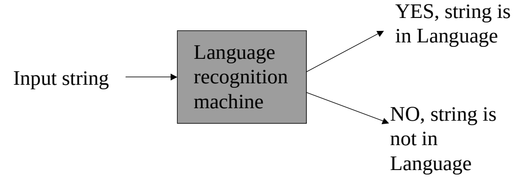
- Recall.
  - What is a language?
  - What is a class of languages?

Languages

- A language is a set of strings.
- A class of languages is nothing more than a set of languages

String Recognition machine

- Given a string and a definition of a language (set of strings), is the string a member of the language?



Regular Languages

- Today we continue looking at our first class of languages: Regular languages
  - Means of defining: Regular Expressions
  - Machine for accepting: Finite Automata

Specifying Languages

- Recall: how do we specify languages?
  - If language is finite, you can list all of its strings.
    - L = {a, aa, aba, aca}
  - Descriptive:
    - $L = \{x \mid n_a(x) = n_b(x)\}$
  - Using basic Language operations
    - $L = \{aa, ab\}^* \cup \{b\}\{bb\}^*$
    - Regular languages are described using this last method

Regular Languages

- A regular language over Σ is a language that can be expressed using only the set operations of
  - Union
  - Concatenation
  - Kleene Star

Kleene Star Operation

- The set of strings that can be obtained by concatenating any number of elements of a language L is called the Kleene Star, L^*

$$L^* = \bigcup_{i=0}^{\infty} L^i = L^0 \cup L^1 \cup L^2 \cup L^3 \cup L^4 \ldots$$

- Note that since, L^* contains L^0, ε is an element of L^*

Regular Expressions

- Regular expressions are the mechanism by which regular languages are described:
  - Take the "set operation" definition of the language and:
    - Replace ∪ with +
    - Replace {} with ()
  - And you have a regular expression

Regular expressions

{ε}	ε
{011}	011
{0,1}	0 + 1
{0, 01}	0 + 01
$\{110\}^*\{0,1\}$	$(110)^*(0+1)$
$\{10, 11, 01\}^*$	$(10 + 11 + 01)^*$
$\{0, 11\}^*(\{11\}^* \cup \{101, \varepsilon\})$	$(0 + 11)^*((11)^* + 101 + \varepsilon)$

Regular Expression

- Recursive definition of regular languages / expression over Σ :
  1. $\varnothing$ is a regular language and its regular expression is $\varnothing$
  2. $\{\varepsilon\}$ is a regular language and ε is its regular expression
  3. For each $a \in \Sigma$, $\{a\}$ is a regular language and its regular expression is a

Regular Expression

4. If L_1 and L_2 are regular languages with regular expressions r_1 and r_2 then
-- $L_1 \cup L_2$ is a regular language with regular expression $(r_1 + r_2)$
-- L_1L_2 is a regular language with regular expression (r_1r_2)
-- L_1^* is a regular language with regular expression (r_1^*)

Only languages obtainable by using rules 1-4 are regular languages.

Regular Expressions

- Some shorthand
  - If we apply precedents to the operators, we can relax the full parenthesized definition:
    - Kleene star has highest precedent
    - Concatenation had mid precedent
    - + has lowest precedent
  - Thus
    - $a + b^*c$ is the same as $(a + ((b^*)c))$
    - $(a + b)^*$ is not the same as $a + b^*$

Regular Expressions

- More shorthand
  - Equating regular expressions.
    - Two regular expressions are considered equal if they describe the same language
    - $1^*1^* = 1^*$
    - $(a + b)^* \neq a + b^*$

Regular Expressions

- Even more shorthand
  - Sometimes you might see in the book:
    - r^n where n indicates the number of concatenations of r (e.g. r^6)
    - r^+ to indicate one or more concatenations of r.
  - Note that this is only shorthand!
  - r^6 and r^+ are <u>not</u> regular expressions.

Regular Expressions

- Important thing to remember
  - A regular expression is <u>not</u> a language
  - A regular expression is used to <u>describe</u> a language.
  - It is incorrect to say that for a language L,
    - $L = (a + b + c)^*$
  - But it's okay to say that L is described by
    - $(a + b + c)^*$

Regular Expressions

- Questions?

Examples of Regular Languages

- All finite languages are regular
  - Can anyone tell me why?

Examples of Regular Languages

- All finite languages are regular
  - A finite language L can be expressed as the union of languages each with one string corresponding to a string in L
  - Example:
    - L = {a, aa, aba, aca}
    - L = $\{a\} \cup \{aa\} \cup \{aba\} \cup \{aca\}$
    - Regular expression: (a + aa + aba + abc)

Examples of Regular Languages

- $L = \{x \in \{0,1\}^* \mid |x| \text{ is even}\}$
  - Any string of even length can be obtained by concatenating strings length 2.
  - Any concatenation of strings of length 2 will be even
  - $L = \{00, 01, 10, 11\}^*$
  - Regular expressions describing L:
    - $(00 + 01 + 10 + 11)^*$
    - $((0 + 1)(0 + 1))^*$

Examples of Regular Languages

- $L = \{x \in \{0,1\}^* \mid x \text{ does not end in } 01\}$
  - If x does not end in 01, then either
    - $|x| < 2$ or
    - x ends in 00, 10, or 11
  - A regular expression that describes L is:
    - $\varepsilon + 0 + 1 + (0 + 1)^*(00 + 10 + 11)$

Examples of Regular Languages

- $L = \{x \in \{0,1\}^* \mid x \text{ contains an odd number of 0s}\}$
  - Express x = yz
  - y is a string of the form $y=1^i01^j$
  - In z, there must be an even number of additional 0s or $z = (01^k01^m)^*$
  - x can be described by $(1^*01^*)(01^*01^*)^*$
  - Questions?

Useful properties of regular expressions

- Commutative
  - $L + M = M + L$
- Associative
  - $(L + M) + N = L + (M + N)$
  - $(LM)N = L(MN)$
- Identities
  - $\varnothing + L = L + \varnothing = L$
  - $\varepsilon L = L \varepsilon = L$
  - $\varnothing L = L \varnothing = \varnothing$

Useful properties of regular expressions

- Distributed
  - $L (M + N) = LM + LN$
  - $(M + N)L = ML + NL$
- Idempotent
  - $L + L = L$

Useful properties of regular expressions

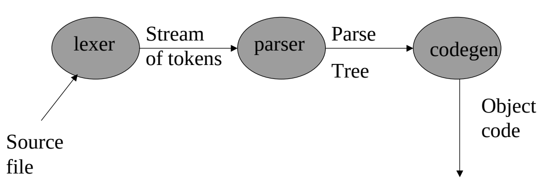
- Closures
  - $(L^*)^* = L^*$
  - $\varnothing^* = \varepsilon$
  - $\varepsilon^* = \varepsilon$
  - $L^+ = LL^*$
  - $L^* = L^+ + \varepsilon$

Practical uses for regular expressions

- grep
  - Global (search for) Regular Expressions and Print
  - Finds patterns of characters in a text file.
  - grep man foo.txt
  - grep [ab]*c[de]? foo.txt

Practical uses for regular expressions

- How a compiler works



Practical uses for regular expressions

- How a compiler works
  - The Lexical Analyzer (lexer) reads source code and generates a stream of tokens
  - What is a token?
    - Identifier
    - Keyword
    - Number
    - Operator
    - Punctuation

Practical uses for regular expressions

- How a compiler works
  - Tokens can be described using regular expressions!

Examples of Regular Languages

- L = set of valid C keywords
  - This is a finite set
  - L can be described by
    - if + then + else + while + do + goto + break + switch + …

Examples of Regular Languages

- L = set of valid C identifiers
  - A valid C identifier begins with a letter or _
  - A valid C identifier contains letters, numbers, and _
  - If we let:
    - l = {a , b , … , z , A , B , … , Z}
    - d = {1 , 2 , … , 9 , 0}
  - Then a regular expression for L:
    - $(l + \_)(l + d + \_)^*$

Practical uses for regular expressions

- lex
  - Program that will create a lexical analyzer.
  - Input: set of valid tokens
  - Tokens are given by regular expressions.

Summary

- Regular languages can be expressed using only the set operations of union, concatenation, Kleene Star.
- Regular languages
  - Means of describing: Regular Expression
  - Machine for accepting: Finite Automata
- Practical uses
  - Text search (grep)
  - Compilers / Lexical Analysis (lex)

- Questions?
- Break time!