CMSC 330: Organization of Programming Languages

Regular Expressions

Reconsider These Three String Operations

- "hello".index("l", 0)
  - return index of the first occurrence of string in s, starting at n
- "hello".sub("h", "j")
  - replace first occurrence of "h" by "j" in string
  - use gsub ("global" sub) to replace all occurrences
- "preprepared".split("re")
  - return array of substrings delimited by occurrences of "re"

- All involve *searching* in a string for a certain pattern
- What if we want to find more complicated patterns?
  - find the first occurrence of "a" or "b"
  - split a string at tabs, spaces, and newlines

Regular Expressions

- A way of describing patterns or sets of strings
  - searching and matching
  - formally describing strings
    - the symbols (lexemes or tokens) that make up a language
- Common to lots of languages and tools
  - UNIX shells; the UNIX utilities awk, sed, perl, grep; languages like Java, OCaml, C libraries, etc.
- Based on some really elegant theory
  - we'll see that soon

Example Regular Expressions in Ruby

- /Ruby/
  - matches the string "Ruby"
  - regular expressions can be delimited by /'s
  - use \ to escape /'s in regular expressions
- /(Ruby|OCaml|Java)/
  - matches either "Ruby", "OCaml", or "Java"
- /(Ruby|Regular)/ or /R(uby|egular)/
  - matches either "Ruby" or "Regular"
  - use ()'s for grouping; use \ to escape ()'s

Using Regular Expressions

- Regular expressions are instances of Regexp
  - but you won't often use its methods
- Basic matching using =~ method of String

```
line = gets()              # read line from standard input
if (line =~ /Ruby/) then   # returns nil if not found
  puts("Found Ruby")
end
```

- Can use regular expressions in index, search, etc.

```
offset = line.index(/(MAX|MIN)/)    # search starting from 0
line.sub(/(Perl|Python)/, "Ruby")   # replace
line.split(/(\t|\n| )/)             # split at tab, space,
                                    # newline
```


Using Regular Expressions, con't.

- Invert matching using !~ method of String
- Matches strings that *don't* contain an instance of the regular expression

Repetition in Regular Expressions

- /(Ruby)*/
  - {"", "Ruby", "RubyRuby", "RubyRubyRuby", ...}
  - * means *zero or more occurrences*
- /Ruby+/
  - {"Ruby", "Rubyy", "Rubyyy", ... }
  - + means *one or more occurrence*
  - so /e+/ is the same as /ee*/
- /(Ruby)?/
  - {"", "Ruby"}
  - ? means *optional*, i.e., zero or one occurrence

Repetition in Regular Expressions, con't.

- /(Ruby){3}/
  - {"RubyRubyRuby"}
  - {x} means repeat the search for **exactly** x occurrences
- /(Ruby){3,}/
  - {"RubyRubyRuby", "RubyRubyRubyRuby", …}
  - {x,} means repeat the search for **at least** x occurrences
- /(Ruby){3, 5}/
  - {"RubyRubyRuby", "RubyRubyRubyRuby", "RubyRubyRubyRubyRuby"}
  - {x, y} means repeat the search for at least x occurrences and at most y occurrences

Watch out for precedence

- /(Ruby)*/ means {"", "Ruby", "RubyRuby", ...}
  - but /Ruby*/ matches {"Rub", "Ruby", "Rubyy", ...}
- In general
  - *, {n}, and + bind most tightly
  - then concatenation (adjacency of regular expressions)
  - then |
- Best to use parentheses to disambiguate

Character Classes

- /[abcd]/
  - {"a", "b", "c", "d"} (Can you write this another way?)
- /[a-zA-Z0-9]/
  - any upper or lower case letter or digit
- /[^0-9]/
  - any character except 0-9
- /[\t\n]/
  - tab, newline or space
- /[a-zA-Z_\$][a-zA-Z_\$0-9]*/
  - Java identifiers ($ escaped...see next slide)

Special Characters

.	any character
^	beginning of string
$	end of string
\$	just a $
\d	digit, [0-9]
\s	whitespace, [\t\r\n\f]
\w	word character, [A-Za-z0-9_]
\D	non-digit, [^0-9]
\S	non-space, [^\t\r\n\f]
\W	non-word, [^A-Za-z0-9_]

Potential Character Class Confusions

- ^
  - at the beginning of a character class: not
  - outside character classes: beginning of string
- [ ]
  - inside regular expressions: character class
  - outside regular expressions: array
    - Note: [a-z] does not make a valid array
- ( )
  - inside character classes: literal characters ( )
    - Note /(0..2)/ does not mean 012
  - outside character classes: used for grouping
- –
  - inside character classes: range (e.g., a to z given by [a-z])
  - outside character classes: a literal minus sign
  - outside regular expressions: subtraction

Regular Expression Practice

- Write Ruby regular expressions representing
  1. All strings beginning with a or b
  2. All strings containing at least two (only alphabetic) words separated by whitespace
  3. All strings where a and b alternate and appear at least once
  4. An expression that would match both of these strings (but not radically different ones)
     - CMSC330: Organization of Programming Languages: Fall 2012
     - CMSC351: Algorithms: Fall 2012