Application and Research Highlights

Regular Expressions

Lecturer : Giorgio Satta

Based on material originally presented in :
https://www.youtube.com/watch?v=Ha9l9fWwO9s

Applications

Pattern matching : text and web search

Lexical analysis : mainly in compilers

Information extraction : extract date & location in emails, data base population

Computational biology : DNA harvester and pattern recognition

Security : search for malicious patterns in stream

grep command

Used in Unix and Emacs

Extends regular expressions with several operators; in most cases, same generative capacity

Several books about grep command !

Example : Operator . is the don't care, use grep for crossword puzzles !

grep command

```
public class Grep
{
  public static void main (String [] args)
  {

    // use NFA: Java class which provides a data type
    // for creating a nondeterministic finite state automaton

    String regex = "(.*" + args[0] + ".*)";  // embedded regex
    NFA nfa = new NFA(regex);

    while ( StdIn.hasNextLine() )
    {
      String line = StdIn.readLine();
      if (nfa.recognizes(line))
        StdOut.println(line);
    }
  }
}
```

Regexp libraries & tools

Many programming languages support extended regular expressions: Awk, Perl, PHP, Python, JavaScript

Many tools for **compiling** regular expressions

Example : Lex and Flex for automatic construction of lexical analyzers

Regexp libraries

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class Harvester
{
  public static void main(String[] args) {
    String regexp = args[0];
    In in = new In(args[1]);  // file or web page
    String input = in.readAll();

    Pattern pattern = Pattern.compile(regexp); // build nfa
    Matcher matcher = pattern.matcher(input); // nfa simulator

    while (matcher.find()) {  // apply simulator
      String s = matcher.group();  // return matching substring
      StdOut.println(s);
    }
  }
}
```

Matching

Reluctant matching closes the match as early as possible

Example : `<blink> .* </blink>`

Greedy matching extends the match as wide as possible

Example : `then` vs. `thenextvalue`

Matching

Back-reference : extension to regex used by several tools

Example : `(.+)\1` implements copying, matching strings of the form *ww* which is not a regular language

the expression between parentheses is called capturing group

With use of back-reference matching becomes intractable

Efficiency

Efficiency may be an issue

Regular expression `(a|aa)*b` might take exponential time in naive implementations, when tested on strings of the form $a^n c$!

Several DoS attack by sending killing email addresses to anti-spam