Monash University
Faculty of Information Technology

Lecture 5
Regular Expressions

Slides by David Albrecht (2011), modified by Graham Farr (2013).

FIT2014 Theory of Computation

Overview

- Some Problems
- Applications of Regular Expressions
- Simple Languages
- Regular Expressions
- Regular Languages

Some Problems

- Find all the files which contain old subject course codes.
- Find all the e-mail addresses in a set of mail files.
- Change the way comments in C programs are formatted in your web pages.
- Using web server access files, record how many times each page is visited, and how many times each link is used.

Applications of Regular Expressions

- Useful way to describe simple patterns.
- Used in several programs:
  - Editors: **vi, emacs**
  - Filters: **egrep, sed, gawk**
  - Programming languages: **JFlex, CUP, Perl**

Filters

- **egrep**
  - A program which searches a file for a pattern described by a regular expression.
- **sed**
  - A program which enables stream editing of files.
- **awk, nawk, gawk**
  - Programming languages which enable text manipulation.

Programming Languages

- **JFlex, flex, lex**
  - Languages used to generate lexical analysers.
- **CUP, bison, yacc**
  - Languages used to generate compilers.
- **Perl**
  - A powerful scripting language, developed in the 1980s by Larry Wall.

Regular Expressions for Small Languages

- Language ϕ with no words:

  ϕ

- Language ε consisting only of the empty word:

  ε

- Language $\{ w \}$ consisting only of the single word w :

  w

- E.g.: the language **{abbab}** consisting only of the single word **abbab :**

  abbab

Alternatives

Alternatives are indicated by ∪ , e.g.

1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9

is a regular expression for:

{1, 2, 3, 4, 5, 6, 7, 8, 9}

Groupings

Groupings are indicated by **( )**, e.g.

(ab ∪ ba)(e ∪ g)

is a regular expression for:

{abe, abg, bae, bag}

This uses the principle that:

if

- R_1 is a regular expression for language L_1, and
- R_2 is a regular expression for language L_2,

then the concatenation R_1R_2 is a regular expression for the language

$\{ x_1x_2 \ : \ x_1 \text{ is in } L_1 \text{ and } x_2 \text{ is in } L_2 \}$

Finite Languages

- consist of finite number of words.
- E.g.

  {abaaba, abbbba, abbaba}

- Regular Expression:

  abaaba ∪ abbbba ∪ abbaba

- alternatively,

  ab(aa ∪ bb ∪ ba)ba

- alternatively,

  ab(a ∪ b)aba ∪ abb(b ∪ a)ba

Kleene Star

- Zero or more times is indicated by *
- For example:

a* represents

{ε, a, aa, aaa, aaaa, …}

(ab)* represents

{ε, ab, abab, ababab, …}

Some infinite languages

- Strings which start with **a** and whose remaining letters (if any) are **b**.

  {a, ab, abb, abbb, abbbb, …}

- Regular Expression

  ab*

  Zero or more

- Note: **ab*** ≠ **(ab)***

(aa ∪ bb)*

(aa ∪ bb)* ← **Zero or more**

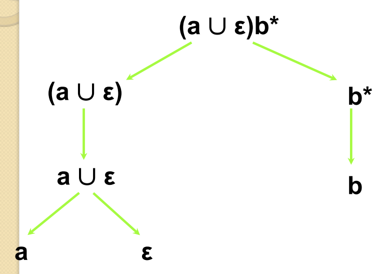
$$= (aa \cup bb)^0 \cup (aa \cup bb)^1 \cup (aa \cup bb)^2 \cup \ldots$$

$$= \varepsilon \cup (aa \cup bb) \cup (aa \cup bb)(aa \cup bb) \cup \ldots$$

represents:

{ε, aa, bb, aaaa, aabb, bbaa, bbbb, aaaaaa, aaaabb, aabbaa, ...}

Parse Tree



Definition

1. ε and ϕ are regular expressions
2. All letters in the alphabet are regular expressions.
3. If **R** and **S** are regular expressions, then so are:
   (i) **(R)**
   (ii) **RS**
   (iii) **R ∪ S**
   (iv) **R***

This is an example of an *inductive definition*, also known as a *recursive definition*.

Regular Language

- A language which can be described by a **Regular Expression** is called a **Regular Language.**
- If a word belongs to the language described by a regular expression, then we say it is **matched** by the regular expression.

Example: EVEN-EVEN

All the strings that contain an even number of **a**'s and an even number of **b**'s.

{ ε , aa, bb, aaaa, aabb, abab, abba, …}

- Regular Expression

  (aa ∪ bb ∪ (ab ∪ ba)(aa ∪ bb)*(ab ∪ ba))*

Things to think about …

Is the set of all English words (in some standard dictionary) a regular language?
Is DOUBLEWORD (see Lecture 1) a regular language?
Is PALINDROME a regular language?
Is the set of all grammatical English sentences a regular language?
How would you determine, for a given string and regular expression, whether the string matches the regular expression?

Example: Floating Point Number

A floating point number has one or more digits, which may begin with a minus sign (-), and which may contain a decimal point.

E.g.

0 1.2 -3 -4.675 002 023.50

Sequence of Digits

- One Digit

  0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9
- Two Digits

  (0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)
- Three Digits

  (0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9) (0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)
- One or more Digits

  (0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)(0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)*

Sequence of Digits

- Digit

  D = (0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)
- Two Digits

  DD or D^2
- Three Digits

  DDD or D^3
- One or more Digits

  DD*

Numbers

- One Digit

  D = (0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9)
- Positive Integers

  N = DD* e.g. **1 123 1209 002 020**
- Integers

  Z = N ∪ (-N)
- Floating Point Number

  F = Z ∪ (Z.) ∪ (.N) ∪ (-.N) ∪ (Z.N)

Other Notations

R | S *means* **R ∪ S**

[0-9] *means*

0 ∪ 1 ∪ 2 ∪ 3 ∪ 4 ∪ 5 ∪ 6 ∪ 7 ∪ 8 ∪ 9

[a-z] *means any letter* **a** *to* **z**

R^+ *means* **RR***

R? *means* **ε ∪ R**

Additional Reading

Jeffrey E.F. Friedl, "***Mastering Regular Expressions: Powerful Techniques for Perl and Other Tools***", O' Reilly, 1997.

Revision

- Regular Expressions
  - Definition.
  - How to use them to define languages

  - read Sipser, section 1.3, pp 63-66

Preparation

- Read

  Sipser, , ***"Introduction to the Theory of Computation"***, Chapter 1.