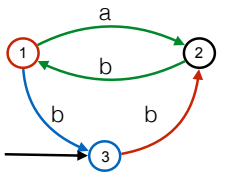
applied regex

CI

- ✓ implementing REs using finite state automata
- • using REs to find patterns

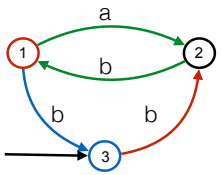
Is there a regular expression for every FSM?



L_3 = (b b (a b)* b)*

L_3 = (b (b a)* b b)*

Is there a regular expression for every FSM?



L_3 = (b b (a b)* b)*

L_3 = (b (b a)* b b)*

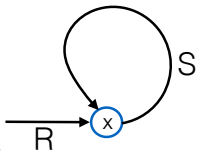
L_3 = ε | b (b b b | b a)* bb

Arden's Lemma

If R and S are
regular expressions
then the equation

X = R | **X** S

has a solution **X** = R S*



If ε ∉ L(S) then this solution is unique.

REs and FSAs

- Regular expressions can be viewed as a textual way of specifying the structure of finite-state automata
- Finite-state automata are a way of implementing regular expressions

Regular Expressions for Textual Searches

Who does it?

Everybody:

- Web search engines, CGI scripts
- Information retrieval
- Word processing (Emacs, vi, MSWord)
- Linux tools (sed, awk, grep)
- Computation of frequencies from corpora
- Perl

Regular Expression

- **Regular expression:** formula in algebraic notation for specifying a set of strings
- **String:** any sequence of alphanumeric characters
  - letters, numbers, spaces, tabs, punctuation marks
- **Regular expression search**
  - **pattern:** specifying the set of strings we want
  - **corpus:** the texts we want to search through

http://www.inf.ed.ac.uk/teaching/courses/inf1/cl/tools/regex-crib.xml

[PDF] Bulk data transfer toolkit: validation rules for CAS ... - Gov…

https://www.gov.uk/.../4__Bulk_Data_Transfer_-_additional_validation_...

If the country code given for the address is 'GBR' then the field is validated against the UK postcode **regular expression**. (see section 3) If the address is given as ...

3. UK postcode regular expression

The following is the UK Postcode Regular Expression and the corresponding detail explaining the logic behind the UK Postcode Regular Expression.

3.1 Expression

^([Gg][Ii][Rr] 0[Aa]{2})|((([A-Za-z][0-9]{1,2})|(([A-Za-z][A-Ha-hJ-Yj-y][0-9]{1,2})|(([AZa-z][0-9][A-Za-z])|([A-Za-z][A-Ha-hJ-Yj-y][0-9]?[A-Za-z])))) [0-9][A-Za-z]{2})$

3.2 Logic

"GIR 0AA"

OR

One letter followed by either one or two numbers

OR

One letter followed by a second letter that must be one of ABCDEFGHJ KLMNOPQRSTUVWXY (i.e..not I) and then followed by either one or two numbers

OR

One letter followed by one number and then another letter

OR

A two part post code

where the first part must be

One letter followed by a second letter that must be one of ABCDEFGH JKLMNOPQRSTUVWXY (i.e..not I) and then followed by one number and optionally a further letter after that

AND

The second part (separated by a space from the first part) must be One number followed by two letters.

A combination of upper and lower case characters is allowed.

Note: the length is determined by the regular expression and is between 2 and 8 characters.

Welcome to RegExLib.com, the Internet's first Regular Expression Library. Currently we have indexed 4736 expressions from 2214 contributors around the world.

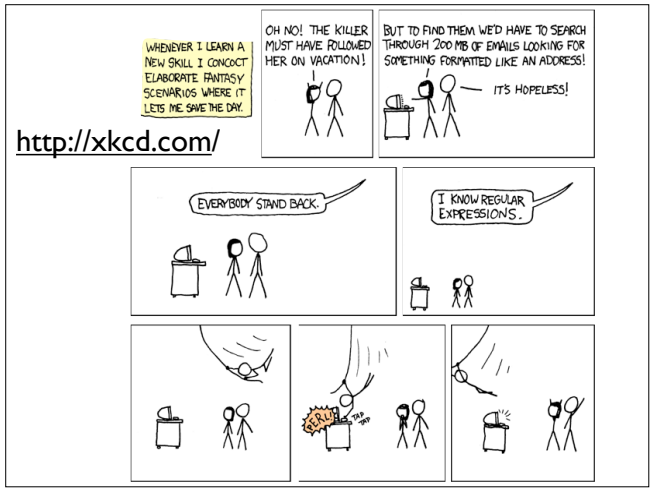
http://www.regexlib.net/

Mastering Regular Expressions

RegEx is supported in all major development environments (for use in editing and working with code) and will thus appeal to anyone using these tools. In addition, every JavaScript developer should be using RegEx, but most don't as it has never been taught to them properly before. Developers using ASP, C#, ColdFusion, Java JSP, PHP, Perl, Python, and more could (and should) be using RegEx.

Oracle Regular Expressions Pocket Reference

Support for regular expressions in SQL and PL/SQL is one of the most exciting features of Oracle Database 10G. Oracle has long supported the ANSI-standard LIKE predicate for rudimentary pattern matching, but regular expressions take pattern matching to a new level. They provide a powerful way to select data that matches a pattern, as well as to manipulate, rearrange, and change that data. This concise pocket guide is part tutorial and part quick-reference.

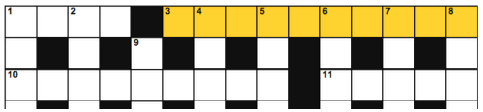


http://xkcd.com/

Everyman crossword No 3,551

Print version | Blind & PS version | PDF version

The Observer, Sunday 26 October 2014 00.00 GMT

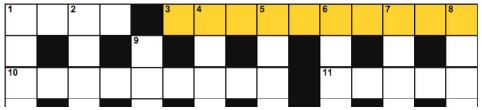


3 Typesetter in awfully poor sitcom (10)

Everyman crossword No 3,551

Print version | Blind & PS version | PDF version

The Observer, Sunday 26 October 2014 00.00 GMT



3 Typesetter in awfully poor sitcom (10)

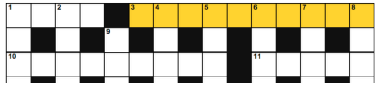
```
% cat /usr/share/dict/words| egrep ^[poorsitcom]{10}$
```

```
$ cat /usr/share/dict/words| egrep ^[poorsitcom]{10}$
compositor
copromisor
crisscross
isoosmosis
isotropism
microtomic
optimistic
poroscopic
postcosmic
postscript
prioristic
promitosis
proproctor
protoprism
tricrotism
troostitic
```

Everyman crossword No 3,551

Print version | Blind & PS version | PDF version

The Observer, Sunday 26 October 2014 00.00 GMT



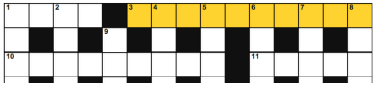
3 Typesetter in awfully poor sitcom (10)

```
% cat /usr/share/dict/words| egrep ^[poorsitcom]{10}$ | grep o.*o.*o
compositor
copromisor
isoosmosis
poroscopic
proproctor
```

Everyman crossword No 3,551

Print version | Blind & PS version | PDF version

The Observer, Sunday 26 October 2014 00.00 GMT



3 Typesetter in awfully poor sitcom (10)

Regular Expressions

- Basic regular expression patterns
- Java-based syntax

- **Disjunctions** `[mM]`

Reg Exp	Match	Example Patterns
`[mM]other`	mother or Mother	"Mother"
`[abc]`	a or b or c	"you are"
`[1234567890]`	any digit	"3 times a day"

Regular Expressions

- **Ranges** `[A-Z]`

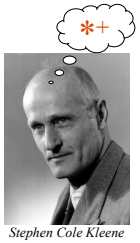
RE	Match	Examples Patterns Matched
[A-Z]	an uppercase letter	"call me Eliza"
[a-z]	a lowercase letter	"call me Eliza"
[0-9]	a single digit	"I'm off at 7"

- **Negations** `[^Ss]`

RE	Match	Examples Patterns Matched
[^A-Z]	not an uppercase letter	"You can call me Eliza"
[^Ss]	neither s nor S	"Say hello Eliza"
[^\.]	not a period	"Hello."

Regular Expressions

- **Optional characters:** `?` ,`*` and `+`
  - `?` (0 or 1)
    `colou?r` → `color` *or* `colour`
  - `*` (0 or more)
    `oo*h!` → `oh!` *or* `ooh!` *or* `ooooh!`
  - `+` (1 or more)
    `o+h!` → `oh!` *or* `ooh!` *or* `ooooh!`



Stephen Cole Kleene

- `.` any char except newline
  `beg.n` → `begin` *or* `began` *or* `begun`

Regular Expressions

- **Anchors ^ and $**
  - ^[A-Z] → "France", "Paris"
  - ^[^A-Z] → "¿verdad?", "really?"
  - \.$ → "It's over."
  - moo$ → "moo", but not "mood"
- **Boundaries \b and \B**
  - \bon\b → "on my way" "Monday"
  - \Bon\b → "automaton"
- **Disjunction |**
  - yours|mine → "it's either yours or mine"

Regular Expressions

http://www.inf.ed.ac.uk/teaching/courses/il1/2010/labs/2010-10-28/regexrepl.xml

- **Replacement**
  - in emacs
  - in javascript
  - in python and perl
  - …

```
s/\bI('m| am)\b /ARE YOU/g
```

- Syntax varies - the ideas are universal

Experiment

http://www.inf.ed.ac.uk/teaching/courses/il1/2010/labs/2010-10-28/regexrepl.xml

- **Replacement**
  - in emacs
  - in javascript
  - in python and perl
  - …

```
s/\bI('m| am)\b /ARE YOU/g
```

- Syntax varies - the ideas are universal