AUTOMATA AND FORMAL LANGUAGES, #course[15103]

REGULAR LANGUAGES, EXPRESSIONS AND APPLICATIONS

DR. VADIM ZAYTSEV A.K.A. @GRAMMARWARE
Roadmap

- Chomsky hierarchy revisited
- How to see if the language is regular?
- The class of regular languages
- Tools to work with regular languages
- Advanced methods
Chomsky: Automata

- Turing machine
- Pushdown automaton
- Linear bounded automaton
- Finite state automaton
- Finite automaton
- Regular language
- Context-free language
- Context-sensitive language
- Recursively enumerable language
- Languages

(too many to list)
Chomsky: Rewriting

\[ \alpha X \beta \rightarrow \alpha \gamma \beta \]

\[ X \rightarrow a \]
\[ X \rightarrow aB \]

Regexp Revisited

- Regular sets by Stephen Kleene in 1956
- $\emptyset$, $\varepsilon$, letters from $\Sigma$
- concatenation
- iteration
- alternation
- Precisely fit the regular class

photo from: Konrad Jacobs, S. C. Kleene, 1978, MFO.
Deterministic Finite Automaton

(finite state grammars and finite diagrams and finite state Markov processes)
To which class do languages belong?

- $\emptyset$ **FINITE**
- $\{\varepsilon\}$ **FINITE**
- $\{\varepsilon\}$ in a non-empty alphabet **FINITE**
- $\{x, y, z\}$ **FINITE**
- $\{0^n | n > 1\}$ **REGULAR**
- Decimal numbers **REGULAR**
- $\{0^n1^n | n > 1\}$ **C-FREE**
- $\{0^n1^2^n | n > 1\}$ **C-FREE**
- $\{0^n1^n2^n | n > 1\}$ **C-SENSITIVE**
**Is a task solvable by regular means?**

- **Substring search**
  - grep, contains(), find(), substring(), …

- **Substring replacement**
  - sed, awk, perl, vim, replace(), replaceAll(), …

- **Pretty-printing**
  - VS.NET, Sublime, TextMate, …

The interactive check marks the following:

- Substring search
- Substring replacement
Is a task solvable by regular means?

- Counting [non-empty] lines in a file
  - `wc -l, grep -c ""`
  - `grep -v "^$", sed -n /./p | wc -l, ...

- Parsing HTML
  - `<BODY><TABLE><P><A HREF=...

- Parsing a postcode
  - `1098 XG, ...

Interactive
HOW TO PROVE WHICH CLASS A LANGUAGE BELONGS TO
PUMPING LEMMA
FOR REGULAR LANGUAGES

- In simple terms
  - sufficiently long words have repeatable parts
  - (works for all infinite regular languages)
- $L$ is regular $\Rightarrow$ formula holds
- Formula does not hold $\Rightarrow L$ is finite or not regular

Jos C.M. Baeten, Models of Computation: Automata, Formal Languages and Communicating Processes, §2.9, p.58.
John Myhill and Anil Nerode
Myhill–Nerode theorem

- Myhill-Nerode equivalence
  
  \[ u \sim v \iff \forall w: (uw \in L \land vw \in L) \lor (uw \notin L \land vw \notin L) \]

- Theorem: L is regular iff the number of Myhill-Nerode equivalence classes is finite.

- In simple terms
  
  - few groups of forgettable prefixes

- Works both ways

LIMITED MEMORY

- Advice from teh internetz:

  - how many characters must you remember from the stream?
    - bounded ⇒ regular
    - unbounded ⇒ ?

- Correct or not?

Number of Counters

- \( \{0^i1^n\ldots\} \)
- no relation between \( i \) and \( n \) \( \Rightarrow \) regular
- 1 counter \( \Rightarrow \) context-free
- \( n \) counters \( \Rightarrow \) context-sensitive
- \( \infty \) counters \( \Rightarrow \) recursively enumerable

Disassemble/massage

- \{0^n1^n \mid n > 1\}
- \{0^i1^n \mid n > 1, i > 1, i \neq n\}

- Matching brackets language not regular
  - \implies no matching pairs language is regular

- Many combinations of regular languages are regular

- Proving by decomposition is valid
THE CLASS OF REGULAR LANGUAGES
Class closed under complement

- If $A$ is a regular language, then
- $\overline{A}$ is regular
- Meaning...
- `grep -v "123" file.txt`
- (Must know the alphabet $\Sigma$)
- (Actually stronger: any finite number of errors)

J. E. Hopcroft, R. Motwani, J. D. Ullman, Introduction to Automata Theory, Languages and Computation, Chapter 4.
**Class closed under set union**

- If $A$ and $B$ are regular languages, then $A \cup B$ is regular.
- Meaning...
  - $[a-z]$
  - $x \mid y \mid z$ (in some notations)

CLASS CLOSED UNDER INTERSECTION

• If A and B are regular languages, then
  • \(A \cap B\) is regular

• Meaning…
  • `cat file.txt | grep "abc" | grep "xyz"`

• (Not true for context-free languages!)

Class closed under difference

- If A and B are regular languages, then
  - \( A \setminus B \) is regular
- Meaning…
  - `cat file.txt | grep "abc" | grep -v "123"
- (Not true for context-free languages!)

CLASS CLOSED UNDER ITERATION

• If $A$ is a regular language, then
  • $A^*$ and $A^+$ are regular

• Meaning…
  • $[a]^*$
  • $[a]^+$

Class closed under concatenation

- If $A$ and $B$ are regular languages, then $AB$ is regular.

- Meaning…
  - $[Bb][Oo][Dd][Yy]$
  - (Just glue regexps; in practice, watch out for subgroups)

CLASS [SOMETIMES] CLOSED UNDER DECOMPOSITION

• If $A$ is a regular language, then
  • “front halves” is regular
  • “tail halves” is regular
  • “middle thirds” is regular
  • “arbitrary halves/thirds” is regular

• NB: glued side thirds is NOT regular

CLASS CLOSED UNDER HOMOMORPHISM

• If A is a regular language and
  • $h : \Sigma \rightarrow \Sigma^*$

• then
  • $h(A)$ is regular

• Meaning that debugging is feasible

• (Even better for context-free languages: substitutions)

var whitelist = 
"</p>|<br\s?/?>|</b>|</strong>|</i>|</em>|</s>|</strike>|</blockquote>|</sub>|</sup>|
</h(1|2|3)>|</pre>|<hr\s?/?>|</code>|</ul>|
</ol>|</li>|</a>|<a[^>]+>|<img[^>]+/?>";

“somewhat pushes the limits of what it is sensible to do with regular expressions”

Jeff Atwood, Regex use vs. Regex abuse, 16 Feb 2005. RFC822.
Paul Warren, Mail::RFC822::Address: regexp-based address validation, 17/09/2012.
TOOLS OVERVIEW
ALL TOOLS ARE DIFFERENT

POSIX standard since 1993

(who the hell uses [:digit:] anyway?)
• Ken Thompson: qed, ed, grep

• grep “abc” program.c

• grep \d file.txt

• grep ^#*\ \w README.md

photo from: Archetypal hackers ken (left) and dmr (right).
SED

- `sed 's/Finite/Regular/g' oldfile >newfile`
- `sed -n 12,18p myfile`
- `sed 12,18d myfile`
- `sed 12q myfile`
- `sed 12,/foo/ d myfile`
- `sed '$d' myfile`
- `sed -n '/[0-9]{2}/p' myfile`
- `sed '5!s/Finite/Regular/g' oldfile >newfile`
- `sed ‘/Automaton/!s/Finite/Regular/g’ oldfile >newfile`

Lee E. McMahon, sed, Stream EDitor, 1973 or 1974, [http://www.columbia.edu/~rh120/ch106.x09](http://www.columbia.edu/~rh120/ch106.x09)

photo from: [http://merdivengo.blogspot.com/2012/03/turnuva-sistemleri-uzerine.html](http://merdivengo.blogspot.com/2012/03/turnuva-sistemleri-uzerine.html)
Origins of AWK

Last login: Sat Apr 26 15:46:02 on ttys010

src/zaytsev$ cd projects/zoo/src/

src/zaytsev$ ls -al

total 8

drwxr-xr-x  6 zaytsev  staff  204 Mar 31  20:57 .
drwxr-xr-x 13 zaytsev  staff  442 Mar 27  22:13 ...
-rw-r--r--  1 zaytsev  staff  281 Mar 26  17:19 Plugin.rsc
drwxr-xr-x 12 zaytsev  staff  408 Apr  9  22:46 framework
drwxr-xr-x  4 zaytsev  staff  136 Mar 26  17:19 javascript
drwxr-xr-x  3 zaytsev  staff  102 Mar 31  21:04 tryout

src/zaytsev$
AWK

- Turing-complete one-liner language with regexps
- Built-in variables
  - $0, NF, $1, $2, $3, ..., $NF
  - FILENAME, NR, FS, OFS, RS, ORS
- Built-in operators
  - print, printf, length, $
- Can define own functions & variables

AWK IN ACTION

Last login: Sat Apr 26 15:46:02 on ttys010

>>>15:46<<<<~ zaytsev$ cd projects/zoo/src/

>>>15:46<<<<src zaytsev$ ls -al

    total 8
    drwxr-xr-x 6 zaytsev staff 204 Mar 31 20:57 .
    drwxr-xr-x 13 zaytsev staff 442 Mar 27 22:13 ..
    lrw-rw-r--  1 zaytsev staff 281 Mar 26 17:19 Plugin.rsc
    drwxr-xr-x 12 zaytsev staff 408 Apr  9 22:46 framework
    drwxr-xr-x  4 zaytsev staff 136 Mar 26 17:19 javascript
    drwxr-xr-x  3 zaytsev staff 102 Mar 31 21:04 tryout

>>>15:46<<<<~ zaytsev$ ls -al | awk '!/total/ {print $6}' | sort - | uniq

Apr
Mar

>>>15:53<<<<~ zaytsev$
**AWK Examples**

- ```
   { 
     w += NF
     c += length + 1 
   }
   END { print NR, w, c }
```  

- ```
   yes Wikipedia | awk 'NR % 4 == 1 { printf "%6d %s\n", NR, $0 }' | sed 5q
```  

- ```
   #!/usr/bin/awk -f
   BEGIN { print "Hello, world!" }
```
awk output

```
19:35$ yes Wikipedia | awk 'NR % 4 == 1 { printf "%6d %s\n", NR, $0 }' | sed 5q
  1 Wikipedia
  5 Wikipedia
  9 Wikipedia
 13 Wikipedia
 17 Wikipedia
```

19:36$
**LEX**

- Regexps used for the first phase of parsing since 1968.
- Wikipedia explains why it is used together with yacc/bison:

```c
(Lex is limited to simple finite state automata).[clarification needed]
```

- Collection of regexp patterns with actions
  
  ```c
  [a-zA-Z]+ { printf("Word: %s\n", yytext); } 
  .\n {}
  ```

- Easy to write a tokeniser


Lex example

```c
int lineno=1;

letter ::= [a-zA-Z]
digit ::= [0-9]
id ::= {letter}({letter}|{digit})*
number ::= {digit}+

%%
printf("\nTokeniser running -- ^D to exit\n");

{id} {line();printf("<id>");}
{id} printf("<id>");
{number} {line();printf("<number>");}
{number} printf("<number>");
^[ \t]+ line();
[ \t]+ printf(" ");
[\n] ECHO;
^[^a-zA-Z0-9 \t\n]+ {line();printf("\%s", yytext);}
[^a-zA-Z0-9 \t\n]+ printf("\%s", yytext);

%%
line()
{
    printf("%4d: ", lineno++);
}
```

M. G. Roth, CS 631, Lex example, https://www.cs.uaf.edu/~cs631/lex_token.txt
Perl [, TCL, Python, ...]

• Henry Spencer made advanced regex in 1986
  • his DFA/NFA-based TCL version is faster!

• Can be used as sed:
  • perl -pi -w -e 's/Perl/Python/g;' *

• Or, in programs:
  • $bar =~ /foo/

• Redesigned in Perl 6 (merged with PEG)
Perl Regex Examples

- **Match**
  
  ```perl
  my ($hs, $ms, $ss) = ($time =~ m/(\d+):(\d+):(\d+)/);
  ```

- **Substitute**
  
  ```perl
  $s =~ s/dog/cat/;
  ```

- **Transliterate**
  
  ```perl
  $uc =~ tr/a-z/A-Z/;
  ```
PCRE

• “Perl Compatible Regular Expressions”
  • P.S.: not compatible with Perl
  • P.P.S.: not regular

• C library by Philip Hazel (stable release Dec. 2013)

• PCREs are used in other languages
  • PHP, Ruby, JavaScript, …

• Way beyond regular: backrefs, recursion, assertions, …
  • <(\w+)>.*<\</1>, \((?R)\)
Rascal (Metaprogramming)

- Java Regex
  - `/xyz/ := “xyz”`
  - if (/xyz/ := s) {...}
  - if (/x<m:y+>z/ := s) println(m);
  - `/\d+ \w*/ := “1098 XG”`

- Lexical grammars
  - lexical Number = [1-9][0-9]*;
  - parse(#Number, file);

http://rascal-mpl.org/
CONCLUSION

• Benefits of regular languages:
  • lexical tools are fast & always applicable
  • (relatively) easy to develop

• Drawbacks:
  • very limited context
  • (usually) many false positives, requires tweaking
Summary

- Chomsky hierarchy
  - languages, automata, algorithms, rewriting systems, hardware
- Judging if regular
  - pumping lemma, Myhill-Nerode, memory, counters, disassembly
- Class closed under
  - complement, union, intersection,
  - difference, concatenation,
  - homomorphism
- Tools
  - grep, perl, sed, awk, rsc
Thanks for your attention!

- This was Dr. Vadim Zaytsev a.k.a. grammarware
  - grammarware.net, twitter.com/grammarware, grammarware.github.com, …
- I usually teach at Master Software Engineering
- and do research on grammars and software languages
- Affiliations
- Slides are CC-BY-SA: grammarware.net/slides/2014/regular.pdf