5.4 Regular Expressions

- regular expressions
- REs and NFAs
- NFA simulation
- NFA construction
5.4 Regular Expressions

- regular expressions
- REs and NFAs
- NFA simulation
- NFA construction
Pattern matching

Substring search. Find a single string in text.

Pattern matching. Find one of a specified set of strings in text.

Ex. [genomics]

- Fragile X syndrome is a common cause of mental retardation.
- A human's genome is a string.
- It contains triplet repeats of CGG or AGG, bracketed by GCG at the beginning and CTG at the end.
- Number of repeats is variable and is correlated to syndrome.

pattern \( GCG(\text{CGG}|\text{AGG})^*CTG \)

text \( GCGCGTGTGTGCGAGAGAGTGGGTTTTAAAGCTG\textcolor{red}{GCGCGAGGCGGCTG}GCGCGGAGGCTG\)
public class NFA
{
    private Digraph G;       // digraph of epsilon transitions
    private String regexp;   // regular expression
    private int M;           // number of characters in regular expression

    // Create the NFA for the given RE
    public NFA(String regexp)
    {
        this.regexp = regexp;
        M = regexp.length();
        Stack<Integer> ops = new Stack<Integer>();
        G = new Digraph(M+1);
        ...
    }
}

input          output
-----          -------
Ada            HTML
Asm            XHTML
Applescript    LATEX
Awk            MediaWiki
Bat            ODF
Bib            TEXINFO
Bison          ANSI
C/C++          DocBook
C#            
Cobol          
Caml          
Changelog      
Css            
D             
Erlang         
Flex           
Fortran        
GLSL           
Haskell        
Html           
Java           
Javalog        
Javascript     
Latex          
Lisp           
Lua            
⋯
grep

A Quick Pocket Reference for a Utility
Every Unix User Needs

grep
Pocket Reference

Detailed Solutions in Eight
Programming Languages

Regular Expressions Cookbook

Mastering
Regular Expressions

John Bambenek &
Agnieszka Klus

O'REILLY®

Jan Goyvaerts & Steven Levithan

O'REILLY®
Regular expressions

A regular expression is a notation to specify a set of strings.

<table>
<thead>
<tr>
<th>operation</th>
<th>order</th>
<th>example RE</th>
<th>matches</th>
<th>does not match</th>
</tr>
</thead>
<tbody>
<tr>
<td>concatenation</td>
<td>3</td>
<td>AABAAB</td>
<td>AABAAB</td>
<td>every other string</td>
</tr>
<tr>
<td>or</td>
<td>4</td>
<td>AA</td>
<td>BAAB</td>
<td>AA BAAB</td>
</tr>
<tr>
<td>closure</td>
<td>2</td>
<td>AB*A</td>
<td>AA ABBBBBBBBBA</td>
<td>AB ABABA</td>
</tr>
<tr>
<td>parentheses</td>
<td>1</td>
<td>A(A</td>
<td>BAAB</td>
<td>AAAAB ABAAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(AB)*A</td>
<td>A ABABABABABABA</td>
<td>AA ABBA</td>
</tr>
</tbody>
</table>
Regular expression shortcuts

Additional operations further extend the utility of REs.

<table>
<thead>
<tr>
<th>operation</th>
<th>example RE</th>
<th>matches</th>
<th>does not match</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>wildcard</strong></td>
<td>.U.U.U.</td>
<td>CUMULUS JUGULUM</td>
<td>SUCCUBUS TUMULTUOUS</td>
</tr>
<tr>
<td><strong>character class</strong></td>
<td>[A-Za-z][a-z]*</td>
<td>word Capitalized</td>
<td>camelCase 4illegal</td>
</tr>
<tr>
<td><strong>one or more</strong></td>
<td>A(BC)+DE</td>
<td>ABCDE ABCBCE</td>
<td>ADE BCDE</td>
</tr>
<tr>
<td><strong>exactly k</strong></td>
<td>[0-9]{5}-[0-9]{4}</td>
<td>08540-1321 19072-5541</td>
<td>1111111111 166-54-111</td>
</tr>
</tbody>
</table>

**Note.** These operations are useful but not essential.

**Ex.** [A-E]+ is shorthand for (A|B|C|D|E)(A|B|C|D|E)*
Regular expression examples

RE notation is surprisingly expressive.

<table>
<thead>
<tr>
<th>regular expression</th>
<th>matches</th>
<th>does not match</th>
</tr>
</thead>
<tbody>
<tr>
<td>.<em>SPB.</em></td>
<td>RASPBERRY CRISPBREAD</td>
<td>SUBSPACE SUBSPECIES</td>
</tr>
<tr>
<td>(substring search)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[0-9]{3}-[0-9]{2}-[0-9]{4}</td>
<td>166-11-4433 166-45-1111</td>
<td>11-55555555 8675309</td>
</tr>
<tr>
<td>(U. S. Social Security numbers)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[a-z]+@[a-z]+.[a-z</td>
<td>com]</td>
<td><a href="mailto:wayne@princeton.edu">wayne@princeton.edu</a> <a href="mailto:rsk@princeton.edu">rsk@princeton.edu</a></td>
</tr>
<tr>
<td>(simplified email addresses)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[$_A-Za-z][_A-Za-z0-9]*</td>
<td>ident3 PatternMatcher</td>
<td>3a ident#3</td>
</tr>
<tr>
<td>(Java identifiers)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

REs play a well-understood role in the theory of computation.
Illegally screening a job candidate

“[First name]! and pre/2 [last name] w/7
bush or gore or republican! or democrat! or charg!
or accus! or criticiz! or blam! or defend! or iran contra
or clinton or spotted owl or florida recount or sex!
or controvers! or fraud! or investigat! or bankrupt!
or layoff! or downsiz! or PNTR or NAFTA or outsourc!
or indict! or enron or kerry or iraq or wmd! or arrest!
or intox! or fired or racis! or intox! or slur!
or controvers! or abortion! or gay! or homosexual!
or gun! or firearm!”

— LexisNexis search pattern used by Monica Goodling
to screen candidates for DOJ positions

Regular expressions to the rescue

http://xkcd.com/208
5.4 Regular Expressions

- regular expressions
- REs and NFAs
- NFA simulation
- NFA construction
Duality between REs and DFAs

**RE.** Concise way to describe a set of strings.

**DFA.** Machine to recognize whether a given string is in a given set.

**Kleene's theorem.**
- For any DFA, there exists a RE that describes the same set of strings.
- For any RE, there exists a DFA that recognizes the same set of strings.

\[
0^* \mid (0^*10^*10^*10^*)^*
\]

number of 1's is a multiple of 3

\[
\begin{align*}
1 & \rightarrow 1 \\
1 & \rightarrow 2 \\
2 & \rightarrow 0
\end{align*}
\]

number of 1's is a multiple of 3

Work by Stephen Kleene in the 1930s!
Pattern matching implementation: basic plan (first attempt)

Overview is the same as for Knuth-Morris-Pratt.
- No backup in text input stream.
- Linear-time guarantee.

Underlying abstraction. Deterministic finite state automata (DFA).

Basic plan. [apply Kleene’s theorem]
- Build DFA from RE.
- Simulate DFA with text as input.

Bad news. Basic plan is infeasible (DFA may have exponential # of states).
Overview is similar to Knuth-Morris-Pratt.

- No backup in text input stream.
- Quadratic-time guarantee (linear-time typical).

Underlying abstraction. Non-deterministic finite state automata (NFA).

Basic plan. [apply Kleene’s theorem]

- Build NFA from RE.
- Simulate NFA with text as input.

Q. What is an NFA?
Nondeterministic finite-state automata

Regular-expression-matching NFA.
- We assume RE enclosed in parentheses.
- One state per RE character (start = 0, accept = \(M\)).
- Red \(\varepsilon\)-transition (change state, but don't scan text).
- Black match transition (change state and scan to next text char).
- Accept if any sequence of transitions ends in accept state.

Nondeterminism.
- One view: machine can guess the proper sequence of state transitions.
- Another view: sequence is a proof that the machine accepts the text.

NFA corresponding to the pattern \(( ( A * B | A C ) D )\)
Nondeterministic finite-state automata

Q. Is A A A A B D matched by NFA?
A. Yes, because some sequence of legal transitions ends in state 11.

NFA corresponding to the pattern ( ( A * B | A C ) D )
Q. Is A A A B D matched by NFA?
A. Yes, because some sequence of legal transitions ends in state 11. 

[ even though some sequences end in wrong state or get stuck ]

NFA corresponding to the pattern \(( ( A * B | A C ) D )\)
Nondeterministic finite-state automata

Q. Is A A A C matched by NFA?
A. No, because no sequence of legal transitions ends in state 11.
   [ but need to argue about all possible sequences ]

NFA corresponding to the pattern ( ( A * B | A C ) D )
Nondeterminism

Q. How to determine whether a string is matched by an automaton?

DFA. Deterministic $\Rightarrow$ easy (only one applicable transition at each step).

NFA. Nondeterministic $\Rightarrow$ hard (can be several applicable transitions at each step; need to select the "right" ones!)

Q. How to simulate NFA?
A. Systematically consider all possible transition sequences. [stay tuned]

NFA corresponding to the pattern ( ( A * B | A C ) D )
5.4 Regular Expressions

- regular expressions
- REs and NFAs
- NFA simulation
- NFA construction
NFA representation

**State names.** Integers from 0 to $M$. 

number of symbols in RE

**Match-transitions.** Keep regular expression in array $re[]$.

<table>
<thead>
<tr>
<th>$re[]$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(</td>
<td>(</td>
<td>A</td>
<td>*</td>
<td>B</td>
<td></td>
<td>A</td>
<td>C</td>
<td>)</td>
<td>D</td>
<td>)</td>
</tr>
</tbody>
</table>

**ε-transitions.** Store in a digraph $G$.

0→1, 1→2, 1→6, 2→3, 3→2, 3→4, 5→8, 8→9, 10→11

NFA corresponding to the pattern ( ( A * B | A C ) D )
**NFA simulation demo**

**Goal.** Check whether input matches pattern.

NFA corresponding to the pattern ( ( A * B | A C ) D )
NFA simulation demo

**Goal.** Check whether input matches pattern.

NFA corresponding to the pattern \(( ( A^* B \mid A C ) D )\)
NFA simulation demo

Read next input character.
- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable from start: 0
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions
NFA simulation demo

Read next input character.
- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable via $\varepsilon$-transitions from start : \{ 0, 1, 2, 3, 4, 6 \}
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by ε-transitions

set of states reachable after matching A
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable after matching A: \{ 3, 7 \}
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable via $\varepsilon$-transitions after matching A
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable via $\varepsilon$–transitions after matching A : { 2, 3, 4, 7 }
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

![Diagram showing state transitions for matching input A A over states A, B, C, D]

Set of states reachable after matching A A
NFA simulation demo

Read next input character.
- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable after matching A A : \{ 3 \}
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable via $\varepsilon$-transitions after matching A A
Read next input character.

• Find states reachable by match transitions.
• Find states reachable by ε-transitions

set of states reachable via ε-transitions after matching A A : { 2, 3, 4 }
NFA simulation demo

Read next input character.
- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

![Diagram showing NFA simulation demo]

set of states reachable after matching A A B
NFA simulation demo

Read next input character.
- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

```
| input | A | A | B | D |
```

set of states reachable after matching $A A B$ : $\{ 5 \}$
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

Input: [A A B D]

Set of states reachable via $\varepsilon$-transitions after matching A A B
Read next input character.

- Find states reachable by match transitions.
- Find states reachable by \( \varepsilon \)-transitions

set of states reachable via \( \varepsilon \)-transitions after matching A A B: \{ 5, 8, 9 \}
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable after matching A A B D
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable after matching A A B D : { 10 }
NFA simulation demo

Read next input character.

- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

Set of states reachable via $\varepsilon$-transitions after matching A A B D
NFA simulation demo

Read next input character.
- Find states reachable by match transitions.
- Find states reachable by $\varepsilon$-transitions

set of states reachable via $\varepsilon$-transitions after matching A A B D: { 10, 11 }
When no more input characters:

- Accept if any state reachable is an accept state.
- Reject otherwise.

set of states reachable: \{ 10, 11 \}
5.4 Regular Expressions

- regular expressions
- REs and NFAs
- NFA simulation
- NFA construction
Building an NFA corresponding to an RE

**States.** Include a state for each symbol in the RE, plus an accept state.

NFA corresponding to the pattern \(( (A * B | A C) D )\)
Building an NFA corresponding to an RE

**Concatenation.** Add match-transition edge from state corresponding to characters in the alphabet to next state.

**Alphabet.** A B C D

**Metacharacters.** ( ) . * |
Building an NFA corresponding to an RE

**Parentheses.** Add $\varepsilon$-transition edge from parentheses to next state.

NFA corresponding to the pattern $( ( A \ast B | A C ) D )$
Building an NFA corresponding to an RE

**Closure.** Add three \( \varepsilon \)-transition edges for each \( * \) operator.

![Diagram](image)

NFA corresponding to the pattern \(( ( A * B \mid A C ) D )\)
Building an NFA corresponding to an RE

2-way or. Add two $\varepsilon$-transition edges for each $|$ operator.

![Diagram showing NFA construction rules for 2-way or](image)

NFA corresponding to the pattern $\left( ( A^* B | A C ) D \right)$
NFA construction: implementation

**Goal.** Write a program to build the $\varepsilon$-transition digraph.

**Challenges.** Remember left parentheses to implement closure and or; remember $|$ symbols to implement or.

**Solution.** Maintain a stack.
- ( symbol: push ( onto stack.
- $|$ symbol: push $|$ onto stack.
- ) symbol: pop corresponding ( and any intervening $|$; add $\varepsilon$-transition edges for closure/or.

NFA corresponding to the pattern $( ( A^* B \mid A C ) D )$
NFA construction demo

( ( A * B | A C ) D )
NFA construction demo

( ( A * B | A C ) D )
NFA construction demo

**Left parenthesis.**
- Add $\varepsilon$-transition to next state.
- Push index of state corresponding to ( onto stack.
NFA construction demo

Left parenthesis.
- Add $\varepsilon$-transition to next state.
- Push index of state corresponding to ( onto stack.
Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add $\epsilon$-transitions if next character is $\ast$. 

NFA construction demo

( ( A $\ast$ B | A C ) D )
NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add $\varepsilon$-transitions if next character is $\ast$.
NFA construction demo

Closure symbol.
- Add $\varepsilon$-transition to next state.
NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add $\varepsilon$-transitions if next character is $\ast$. 

\[
( ( A * B \mid A C ) D )
\]
NFA construction demo

Or symbol.
- Push index of state corresponding to | onto stack.
Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add $\varepsilon$-transitions if next character is $\ast$.

\[( ( A \ast B | A C ) D )\]
NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add \( \varepsilon \)-transitions if next character is *.

\[
( ( A \ast B \mid A C ) D )
\]
Right parenthesis.

- Add $\varepsilon$-transition to next state.
- Pop corresponding ( and any intervening |; add $\varepsilon$-transition edges for or.
- Do one-character lookahead:
  add $\varepsilon$-transitions if next character is *.
NFA construction demo

Alphabet symbol.

- Add match transition to next state.
- Do one-character lookahead:
  add $\varepsilon$-transitions if next character is $\ast$. 

$( ( A \ast B \mid A C ) D )$
Right parenthesis.

- Add \( \varepsilon \)-transition to next state.
- Pop corresponding ( and any intervening |;
  add \( \varepsilon \)-transition edges for or.
- Do one-character lookahead:
  add \( \varepsilon \)-transitions if next character is \(*\).
End of regular expression.

- Add accept state.
NFA construction demo

NFA corresponding to the pattern ( ( A * B | A C ) D )
NFA construction demo

NFA corresponding to the pattern ( ( A * B | A C ) D )
Abstract machines, languages, and nondeterminism.
- Basis of the theory of computation.
- Intensively studied since the 1930s.
- Basis of programming languages.

Compiler. A program that translates a program to machine code.
- KMP string ⇒ DFA.
- grep RE ⇒ NFA.
- javac Java language ⇒ Java byte code.
Summary of pattern-matching algorithms

**Programmer.**
- Implement substring search via DFA simulation.
- Implement RE pattern matching via NFA simulation.

**Theoretician.**
- RE is a compact description of a set of strings.
- NFA is an abstract machine equivalent in power to RE.
- DFAs, NFAs, and REs have limitations.

**You.**
- Core CS principles provide useful tools that you can exploit now.
- REs and NFAs provide introduction to theoretical CS.

**Example of essential paradigm in computer science.**
- Build the right intermediate abstractions.
- Solve important practical problems.