Regular Expressions

Reference: Chapter 7.1-7.4, *Introduction to Computer Science*, R. Sedgewick and K. Wayne.

Robert Sedgewick and Kevin Wayne • Copyright © 2005 • http://www.Princeton.EDU/~cos226

Pattern Matching

String search. Search for given string in a large text file.

Regular expression.
- Natural and compact way to express multiple text patterns.
- Quintessential programmer's tool.

Ex. Fragile X syndrome is a common cause of mental retardation.
- Human genome contains triplet repeats of `CGG` or `AGG`, starting with `GCG` and ending with `CTG`.
- Number of repeats is variable, and correlated with syndrome.
- Use regular expression to specify pattern: `GCG(CGG|AGG)*CTG`.

Pattern Matching Applications

Test if a string matches some pattern.
- Process natural language.
- Scan for virus signatures.
- Search for information using Google.
- Access information in digital libraries.
- Retrieve information from Lexis/Nexis.
- Search-and-replace in a word processors.
- Filter text (spam, NetNanny, Carnivore, malware).
- Validate data-entry fields (dates, email, URL, credit card).
- Search for markers in human genome using PROSITE patterns.

Parse text files.
- Compile a Java program.
- Crawl and index the Web.
- Read in data stored in ad hoc input file format.
- Automatically create Java documentation from Javadoc comments.

Regular Expressions: Basic Operations

Regular expression. Notation to specify a set of strings.

Operation	Regular Expression	Yes	No
Concatenation	`aabaab`	`aabaab`	every other string
Wildcard	`.u.u.u.`	`cumulus` `jugulum`	`succubus` `tumultuous`
Union	`aa | baab`	`aa` `baab`	every other string
Closure	`ab*a`	`aa` `abbba`	`ab` `ababa`
Parentheses	`a(a|b)aab`	`aaaab` `abaab`	every other string
	`(ab)*a`	`a` `abababababa`	`aa` `abbba`

Regular Expressions: Examples

Regular expression. Notation is surprisingly expressive.

Regular Expression	Yes	No
`.*spb.*` contains the trigraph `spb`	`raspberry` `crispbread`	`subspace` `subspecies`
`a*	(a*ba*ba*ba*)*` multiple of three `b`'s	`bbb` `aaa` `bbbaababbaa`	`b` `bb` `baabbbaa`
`.*0....` fifth to last digit is `0`	`1000234` `98701234`	`111111111` `403982772`

`gcg(cgg|agg)*ctg`
fragile X syndrome indicator
	`gcgctg`
	`gcgcggctg`
	`gcgcggaggctg`
		`gcgcgg`
		`cggcggcggctg`
		`gcgcaggctg`

Generalized Regular Expressions

Generalized regular expressions.

- Additional operations typically added for convenience.
- Ex: `[a-e]+` is shorthand for `(a|b|c|d|e)(a|b|c|d|e)*`.

Operation	Regular Expression	Yes	No
One or more	`a(bc)+de`	`abcde` `abcbcde`	`ade` `bcde`
Character classes	`[A-Za-z][a-z]*`	`word` `Capitalized`	`camelCase` `4illegal`
Exactly k	`[0-9]{5}-[0-9]{4}`	`08540-1321` `19072-5541`	`111111111` `166-54-111`
Negations	`[^aeiou]{6}`	`rhythm`	`decade`

Regular Expressions in Java

Validity checking. Is `input` in the set described by the `re`?

```
public class Validate {
   public static void main(String[] args) {
      String re    = args[0];
      String input = args[1];
      System.out.println(input.matches(re));
   }
}
```

```
% java Validate "..oo..oo." bloodroot
true

% java Validate "[$_A-Za-z][$_A-Za-z0-9]*" ident123
true

% java Validate "[a-z]+@([a-z]+\.)+(edu|com)" rs@cs.princeton.edu
true

% java Validate "[0-9]{3}-[0-9]{2}-[0-9]{4}" 166-11-4433
true
```

need help solving crosswords?

legal Java identifier

valid email address (simplified)

Social Security number

Regular Expressions in Other Languages

Broadly applicable programmer's tool.

- Many languages support extended regular expressions.
- Built into grep, awk, emacs, Perl, PHP, Python, JavaScript.

```
grep NEWLINE */*.java
```

print all lines containing `NEWLINE` which occurs in any file with a `.java` extension

```
egrep '^[qwertyuiop]*[zxcvbnm]*$' dict.txt | egrep '..........'
```

PERL. Practical Extraction and Report Language.

```
perl -p -i -e 's|from|to|g' input.txt
```

replace all occurrences of `from` with `to` in the file `input.txt`

```
perl -n -e 'print if /^[A-Za-z][a-z]*$/' dict.txt
```

do for each line

Regular Expression Caveats

Writing a RE is like writing a program.

- Need to learn syntax.
- Can be easier to write than read.

> "Sometimes you have a programming problem and it seems like the best solution is to use regular expressions; now you have two problems."

Perl RE for Valid RFC822 Email Addresses

Reference: http://www.ex-parrot.com/~pdw/Mail-RFC822-Address.html

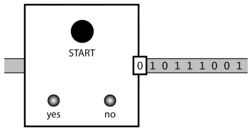
Engineering Grep

Generalized regular expression print.

- First implemented in 1973 by Ken Thompson for text-to-speech.
- Quintessential programmer's tool.

Approach to develop grep algorithm.

- Define class of abstract machines.
- Write simulator for machine.
- Write translator from REs to machines.



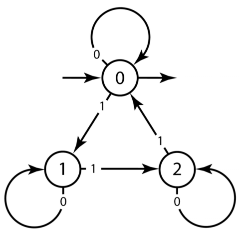
An abstract machine

Example of essential paradigm in computer science.

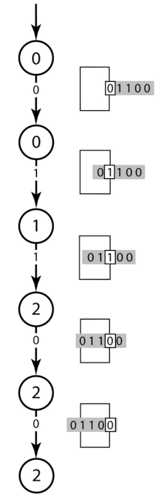
- Build intermediate abstractions.
- Pick the right ones!
- Solve important practical problem.

Deterministic Finite State Automata

DFA review.



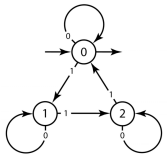

```
int pc = 0;
while (!tape.isEmpty()) {
   boolean bit = tape.read();
   if      (pc == 0) { if (!bit) pc = 0; else pc = 1; }
   else if (pc == 1) { if (!bit) pc = 1; else pc = 2; }
   else if (pc == 2) { if (!bit) pc = 2; else pc = 0; }
}
if (pc == 0) System.out.println("accepted");
else         System.out.println("rejected");
```



Duality

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* | (0*10*10*10*)*
```

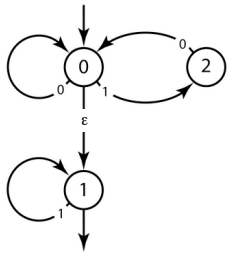
multiple of three 1's

Good news. To match RE, build DFA and simulate DFA on input string.
Bad news. The DFA can be exponentially large.
Consequence. Need more efficient abstract machine.

Nondeterministic Finite State Automata

NFA.

- Finite state automata.
- May have 0, 1, or more transitions for each input symbol.
- May have ε-transitions.
- Accept if any sequence of transitions leads to accept state.

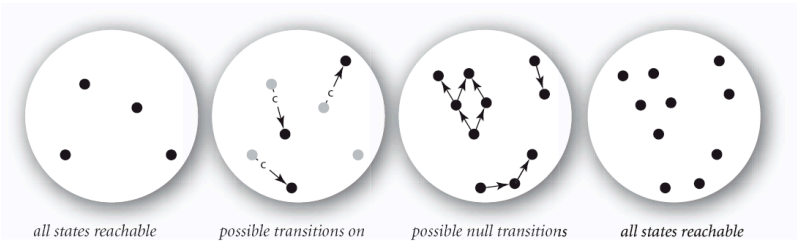


yes: 111, 00011, 101001011

no: 110, 00011011, 00110

Simulating an NFA

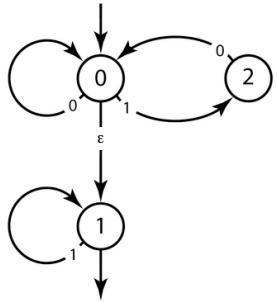
How to simulate an NFA? Maintain set of all possible states that NFA could be in after reading in the first i symbols.

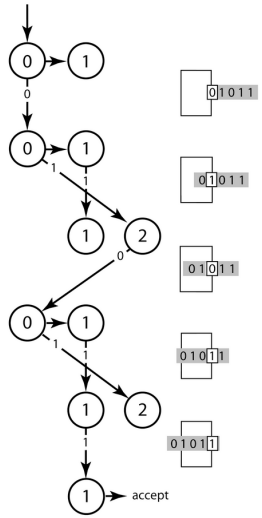


all states reachable after reading i symbols | *possible transitions on reading (i+1)st symbol c* | *possible null transitions before reading next symbol* | *all states reachable after reading i+1 symbols*

One step in simulating an NFA

NFA Simulation

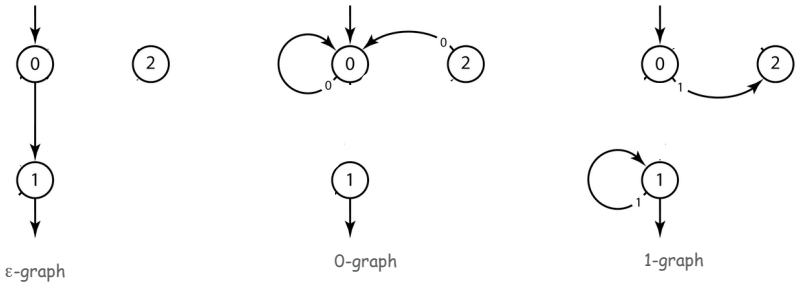




An NFA trace

NFA Representation

NFA representation. Maintain several digraphs, one for each symbol in the alphabet, plus one for ε.



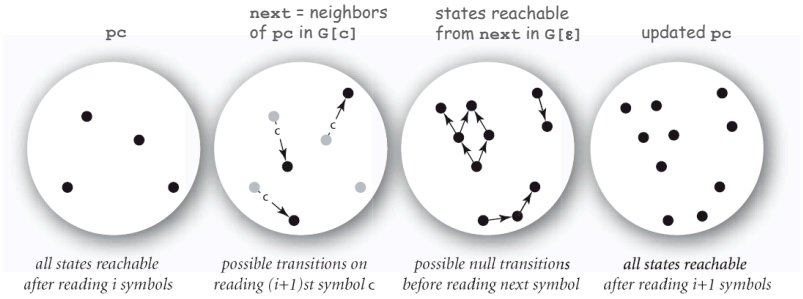
NFA: Java Implementation

```
public class NFA {
   private int START  = 0;               // start state
   private int ACCEPT = 1;               // accept state
   private int N      = 2;               // number of states
   private String ALPHABET = "01";       // RE alphabet
   private int EPS = ALPHABET.length(); // symbols in alphabet
   private Digraph G[];

   public NFA(String re) {
      G = new Digraph[EPS + 1];
      for (int i = 0; i <= EPS; i++)
         G[i] = new Digraph();
      build(0, 1, re);
   }

   private void build(int from, int to, String re) { }
   public boolean simulate(Tape tape)              { }
```


NFA Simulation

How to simulate an NFA?

- Maintain a `SET` of all possible states that NFA could be in after reading in the first i symbols.
- Use `Digraph` adjacency and reachability ops to update.



NFA Simulation: Java Implementation

```
public boolean simulate(Tape tape) {
   SET<Integer> pc = G[EPS].reachable(START);

   // simulate NFA using input from tape
   while (!tape.isEmpty()) {
      char c = tape.read();
      int  i = ALPHABET.indexOf(c);
      SET<Integer> next = G[i].neighbors(pc);

      pc = G[EPS].reachable(next);
   }

   for (int state : pc)
      if (state == ACCEPT) return true;
   return false;
}
```

states reachable from start by ε-transitions

all possible states after reading in c

follow ε-transitions

check if end in an accept state

NFA Simulation Running Time

Input: Text with N characters, NFA with M transitions.

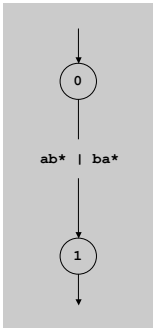
Running time. O(M N)

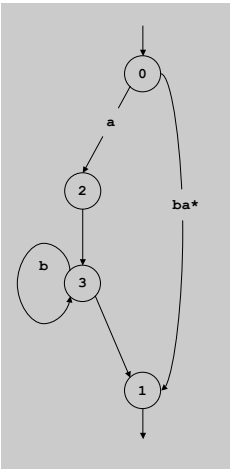
- Bottleneck = 1 graph reachability per input character.
- Can be substantially faster in practice if few ε-transitions.

Note. Easy to extend graph search to handle multiple sources.
Implicit assumption. Alphabet size is a small constant.

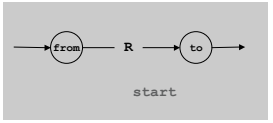
Extended NFA

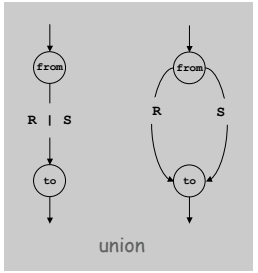
Some extended NFAs.

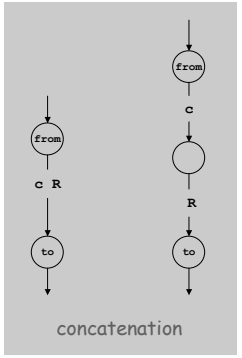


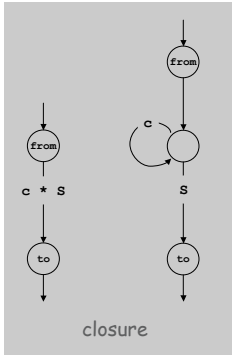


Converting from an RE to an NFA: Basic Transformations



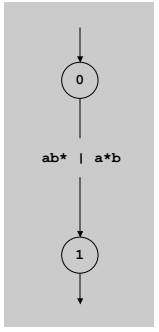


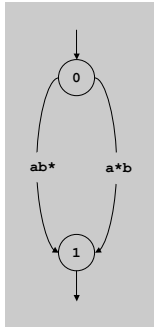


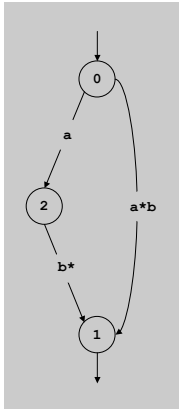


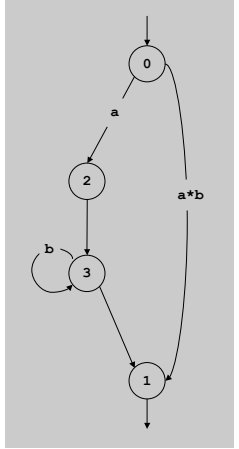
Converting from an RE to an NFA

Ex. Create NFA for `ab* | a*b`.



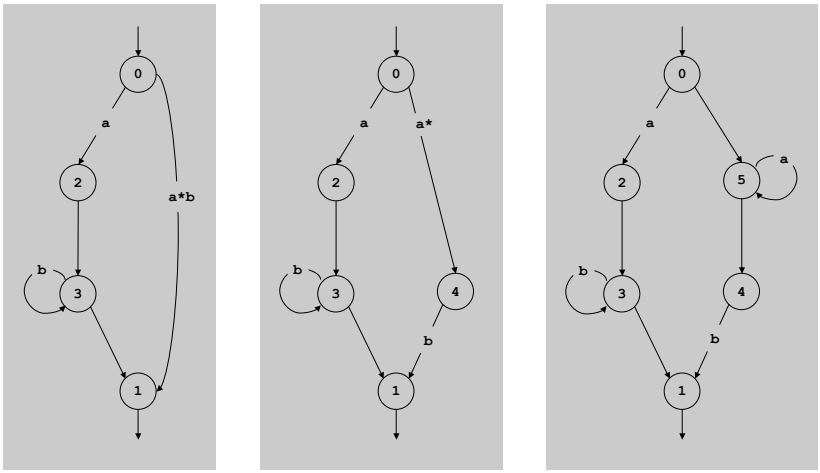






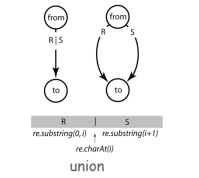
Converting from an RE to an NFA

Ex. Create NFA for `ab* | a*b`.

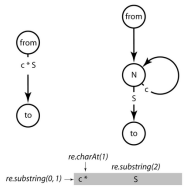


NFA Construction: Java Implementation

```
private void build(int from, int to, String re) {
   int or = re.indexOf('|');

   if (re.length() == 0) G[EPSILON].addEdge(from, to);

   else if (re.length() == 1) {                          single char
      char c = re.charAt(0);
      for (int i = 0; i < EPSILON; i++)
         if (c == ALPHABET.charAt(i) || c == '.')
            G[i].addEdge(from, to);
   }

   else if (or != -1) {                                  union
      build(from, to, re.substring(0, or));
      build(from, to, re.substring(or + 1));
   }

   else if (re.charAt(1) == '*') {                       closure
      G[EPSILON].addEdge(from, N);
      build(N, N, re.substring(0, 1));
      build(N++, to, re.substring(2));
   }

   else {                                                concatenation
      build(from, N, re.substring(0, 1));
      build(N++, to, re.substring(1));
   }
}
```



union



closure

Grep Running Time

Input. Text with N characters, RE with M characters.

Claim. The number of edges in the NFA is at most 2M.

- Single character: consumes 1 symbol, creates 1 edge.
- Wildcard character: consumes 1 symbol, creates 2 edges.
- Concatenation: consumes 1 symbols, creates 0 edges.
- Union: consumes 1 symbol, creates 1 edges.
- Closure: consumes one symbol, creates 2 edges.

NFA simulation. $O(MN)$ since NFA has $2M$ transitions.
NFA construction. Ours is $O(M^2)$ but not hard to make $O(M)$.

Surprising bottom line. Worst case cost for grep is the same as for elementary string match!

Industrial Strength Grep Implementation

To complete grep implementation.

- Parentheses.
- Documentation.
- Extend the alphabet.
- Add character classes.
- Add capturing capabilities.
- Deal with meta characters.
- Extend the closure operator.
- Error checking and recovery.
- Greedy vs. reluctant matching.

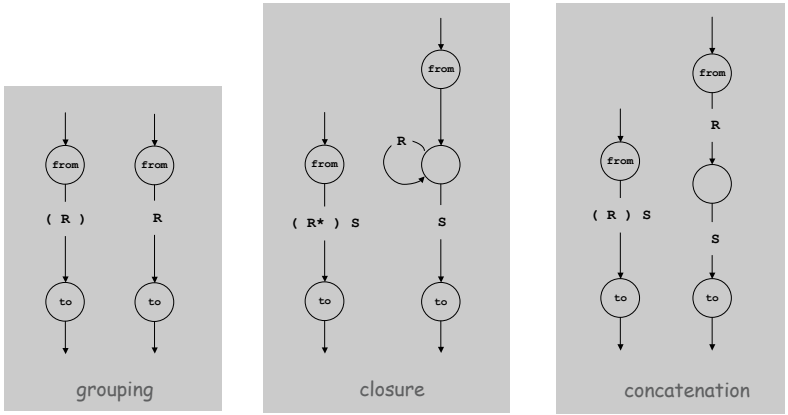
case insensitive reluctant matching

```
String regexp = "(?i)(<blink>)(.+?)(</blink>)";
String update = s.replaceAll(regexp, "$2");
```

remove all blink tags from web page capturing group

Converting from an RE to an NFA

Transformations for parsing parentheses.



Application: Harvester

Harvesting info. Print all occurrences of `regexp` from `text` file or URL.

Pattern. Compiles RE to an NFA.
Matcher. Simulate the NFA.

```
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]);
      String text     = in.readAll();
      Pattern pattern = Pattern.compile(regexp);
      Matcher matcher = pattern.matcher(text);
      while (matcher.find())
         System.out.println(matcher.group());
   }
}
```

find next subsequence of input that matches the pattern

the input subsequence matched by most recent call to find

Application: Harvester

Harvesting info. Print all occurrences of `regexp` from `text` file or URL.

- Harvest patterns from DNA.

```
% java Harvester "gcg(cgg|agg)*ctg" chromosomeX.txt
gcgcggcggcggcggctg
gcgctg
gcgctg
gcgcggcggcggaggcggaggcggctg
```

- Harvest links from website.

```
% java Harvester "http://(\\w+\\.)*(\\w+)" http://www.cs.princeton.edu
http://www.princeton.edu
http://www.google.com
```


Application: Parsing a Data File

Parsing input files. NCBI genome file, ...

```
LOCUS AC146846 128142 bp DNA linear HTG 13-NOV-2003
DEFINITION Ornithorhynchus anatinus clone CLM1-393H9,
ACCESSION AC146846
KEYWORDS HTG; HTGS_PHASE2; HTGS_DRAFT.
SOURCE Ornithorhynchus anatinus (platypus)
ORIGIN
     1 tgtatttcat ttgaccgtgc tgttttttcc cggtttttca gtacggtgtt agggagccac
    61 gtgattctgt ttgttttatg ctgccgaata gctgctcgat gaatctctgc atagacagct  // a comment
   121 gccgcaggga gaaatgacca gtttgtgatg acaaaatgta ggaaagctgt ttcttcataa
   ...
128101 ggaaatgcga cccccacgct aatgtacagc ttctttagat tg
```

```
String regexp   = "[ ]*[0-9]+([actg ]*).*";
Pattern pattern = Pattern.compile(regexp);
In in = new In(filename);
while (!in.isEmpty()) {
   String line = in.readLine();
   Matcher matcher = pattern.matcher(line);
   if (matcher.find()) {
      String s = matcher.group(1).replaceAll(" ", "");
      // do something with s
   }
}
```

the part of the match delimited by the first group of parentheses

replace this RE

with this string

Algorithmic Complexity Attacks

Warning. Most implementations do not guarantee performance!

grep, Java, Perl

```
java Validate "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaac              1.6 seconds
java Validate "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaac             3.7 seconds
java Validate "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac            9.7 seconds
java Validate "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac          23.2 seconds
java Validate "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac         62.2 seconds
java Validate "(a|aa)*b" aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaac 161.6 seconds
```

SpamAssassin regular expression.

```
java RE "[a-z]+@[a-z]+([a-z\.]+\.)+[a-z]+" spammer@x.....................
```

- Takes exponential time.
- Spammer can use a pathological email addresses to DOS a mail server.

Not-So-Regular Expressions

Back-references.

- `\1` notation matches sub-expression that was matched earlier.
- Supported by most RE implementations.

```
java Harvester "^(.*)\1$"  dictionary.txt
beriberi
couscous
```

Some non-regular languages.

- All strings of the form ww for some string w: `beriberi`.
- All bitstring with an equal number of 0s and 1s: `01110100`.
- All Watson-Crick complemented palindromes: `atttcggaaat`.

Remark. Pattern matching with back-references is intractable.

Context

Abstract machines, languages, and nondeterminism.

- Basis of the theory of computation.
- Intensively studied since the 1930s.

Compiler. A program that translates from one language to another.

- `grep` RE ⇒ NFA.
- `javac` Java language ⇒ Java byte code.

Abstraction	RE	Java program
pattern	string	string
parser	check if legal RE	check if legal Java program
compiler	output NFA	output Java byte code
simulator	use NFA to find match	execute byte code on JVM

Summary

Programmer.

- REs are a powerful pattern matching tool.
- Implement regular expressions with NFAs.

Theoretician.

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

You. Practical application of core CS principles.