

7: Theory of Computation

Two fundamental questions.

- What can a computer do?
- What can a computer do with limited resources?

General approach.

- Don't talk about specific machines or problems.
- Consider minimal abstract machines.
- Consider general classes of problems.

Pentium IV running Linux kernel 2.4.22

Why Learn Theory

In theory . . .

- Deeper understanding of what is a computer and computing.
- Foundation of all modern computers.
- Pure science.
- Philosophical implications.

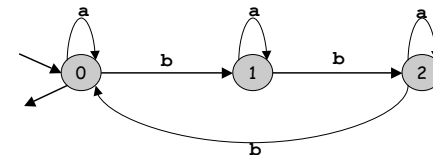
In practice . . .

- Web search: theory of pattern matching.
- Sequential circuits: theory of finite state automata.
- Compilers: theory of context free grammars.
- Cryptography: theory of computational complexity.
- Data compression: theory of information.

"In theory there is no difference between theory and practice. In practice there is." -Yogi Berra

Regular Expressions and DFAs

$a^* \mid (a^*ba^*ba^*ba^*)^*$



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 TOY input file format.
- Automatically create Java documentation from Javadoc comments.

5

Regular Expressions: Examples

Regular expression. Notation is surprisingly expressive.

Regular Expression	Yes	No
<code>. * spb . *</code> contains the trigraph <code>spb</code>	raspberry crispbread	subspace subspecies
<code>a * (a * ba * ba * ba *) *</code> multiple of three b's	bbb aaa bbbaababbaa	b bb baabbaa
<code>. * 0</code> fifth to last digits is 0	10000 98701234	11111111 403982772
<code>gcg (cgg agg) * ctg</code> fragile X syndrome indicator	gcgctg gcgcggctg gcgcggaggctg	gcgcgg cggcggcggctg gcgcggaggctg

7

Regular Expressions: Basic Operations

Regular expression. Notation to specify a set of strings.

Operation	Regular Expression	Yes	No
Concatenation	<code>aabaab</code>	aabaab	every other string
Wildcard	<code>.u.u.u.</code>	cumulus jugulum	succubus tumultuous
Union	<code>aa baab</code>	aa baab	every other string
Closure	<code>ab * a</code>	aa abbba	ab ababa
Parentheses	<code>a (a b) aab</code>	aaaab abaab	every other string
	<code>(ab) * a</code>	a ababababa	ϵ abbbaa

6

Generalized Regular Expressions

Regular expressions are a standard programmer's tool.

- Built in to Java, Perl, Unix, Python,
- 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	<code>a (bc) + de</code>	abcde abcbcede	ade bcde
Character classes	<code>[A-Za-z] [a-z] *</code>	capitalized Word	camelCase 4illegal
Exactly k	<code>[0-9] {5} - [0-9] {4}</code>	08540-1321 19072-5541	11111111 166-54-111
Negations	<code>[^aeiou] {6}</code>	rhythm	decade

8

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));
    }
}
```

↑
powerful string library method

```
% java RE "...oo..oo." bloodroot
true
% java RE "[$_A-Za-z][$_A-Za-z0-9]*" ident123
true
% java RE "[a-z]+@[a-z]+\.(edu|com)" rs@cs.princeton.edu
true
```

need help solving crosswords?
legal Java identifier
valid email address (simplified)
need quotes to "escape" the shell

9

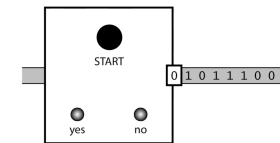
Solving the Pattern Match Problem

Regular expressions are a concise way to describe patterns.

- How would you implement `String.matches`?
- Hardware: build a deterministic finite state automaton (DFA).
- Software: simulate a DFA.

DFA: simple machine that solves the pattern match problem.

- Different machine for each pattern.
- Accepts or rejects string specified on input tape.
- Focus on `true` or `false` questions for simplicity.

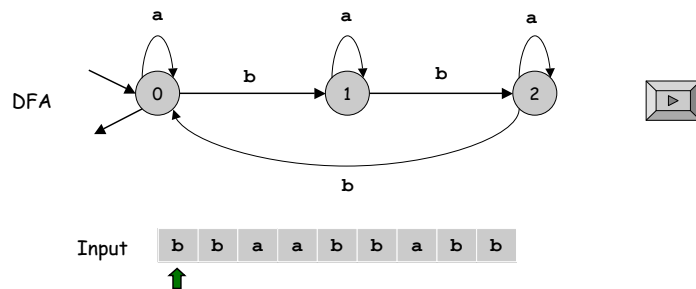


10

Deterministic Finite State Automaton (DFA)

Simple machine with N states.

- Begin in *start state*.
- Read first input symbol.
- Move to new state, depending on current state and input symbol.
- Repeat until last input symbol read.
- Accept or string if unlabeled arc leaves last state.



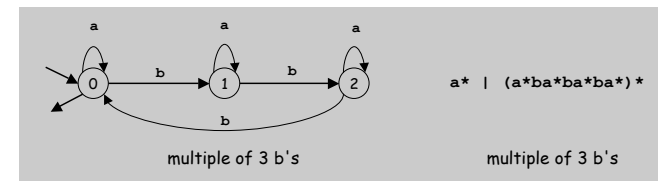
11

Theory of DFAs and REs

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

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

Duality: for any DFA, there exists a regular expression to describe the same set of strings; for any regular expression, there exists a DFA that recognizes the same set.



Practical consequence of duality proof: to match regular expression patterns, (i) build DFA and (ii) simulate DFA on input string.

12

Implementing a Pattern Matcher

Problem: given a regular expression, create program that tests whether given input is in set of strings described.

Step 1: build the DFA.

- A compiler!
- See COS 226 or COS 320.

Step 2: simulate it with given input. Easy.

```
State state = start;
while (!CharStdIn.isEmpty()) {
    char c = CharStdIn.readChar();
    state = state.next(c);
}
System.out.println(state.accept());
```

Application: Harvester

Harvest information from input stream.

- Use `Pattern` data type to compile regular expression to NFA.
- Use `Matcher` data type to simulate NFA.

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;
public class Harvester {
    public static void main(String[] args) {
        String re = args[0];
        In in = new In(args[1]);
        String input = in.readAll();
        Pattern pattern = Pattern.compile(re);
        Matcher matcher = pattern.matcher(input);
        while (matcher.find()) {
            System.out.println(matcher.group());
        }
    }
}
```

13

Application: Harvester

Harvest information from input stream.

- Harvest patterns from DNA.

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

- Harvest email addresses from web for spam campaign.

```
% java Harvester "[a-z]+@[a-z]+\.(edu|com|net|tv)"
http://www.princeton.edu/~cos126
rs@cs.princeton.edu
dgabai@cs.princeton.edu
wayne@cs.princeton.edu
↑
email validator (simplified)
```

14

Application: Parsing a Data File

Ex: parsing an NCBI genome data file.

```
LOCUS AC146846 128142 bp DNA linear HTG 13-NOV-2003
DEFINITION Ornithorhynchus anatinus clone CLM1-393H9,
ACCESSION AC146846
VERSION AC146846.2 GI:38304214
KEYWORDS HTG; HTGS_PHASE2; HTGS_DRAFT.
SOURCE Ornithorhynchus anatinus (platypus)
ORIGIN
1 tgtatttcac ttgaccgtgc tgttttttcc oggttttcca gtaccggtgt agggagccac
61 gtgattctgt ttgttttatg ctgcccgaata gctgctcgat gaatctctgc atagacagct // a comment
121 gccgcagggg gaattgacca gtttgtgatg acaaaatgta ggaagctgt ttcttcataa
...
128101 ggaaatgccg cccccaagct aatgtacagc ttctttagat tg
//
```



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

← extract the RE part in parentheses
↑ ↑
replace this RE with this string

15

16

Fundamental Questions

Which languages CANNOT be described by any RE?

- Bit strings with equal number of 0s and 1s.
- Decimal strings that represent prime numbers.
- Genomic strings that are Watson-Crick complemented palindromes.
- Many more. . . .

How can we extend REs to describe richer sets of strings?

- Context free grammar (e.g., Java).

Reference: http://java.sun.com/docs/books/jls/second_edition/html/syntax.doc.html

Q. How can we make simple machines more powerful?

Q. Are there any limits on what kinds of problems machines can solve?

Summary

Programmer.

- Regular expressions are a powerful pattern matching tool.
- Implement regular expressions with finite state machines.

Theoretician.

- Regular expression is a compact description of a set of strings.
- DFA is an abstract machine that solves pattern match problem for regular expressions.
- DFAs and regular expressions have limitations.

You. Practical application of core CS principles.