Theory of Computation

Introduction to Theoretical CS

Q. What can a computer do?

Q. What can a computer do with limited resources?

General approach.

e.g., Intel Core 2 Duo running Linux kernel 2.6

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

Pioneering work in the 1930s.

- Princeton == center of universe.
- Automata, languages, computability, universality, complexity, logic.

David Hilbert

Kurt Gödel

Alan Turing

Alonzo Church

John von Neumann

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

Describing a Pattern

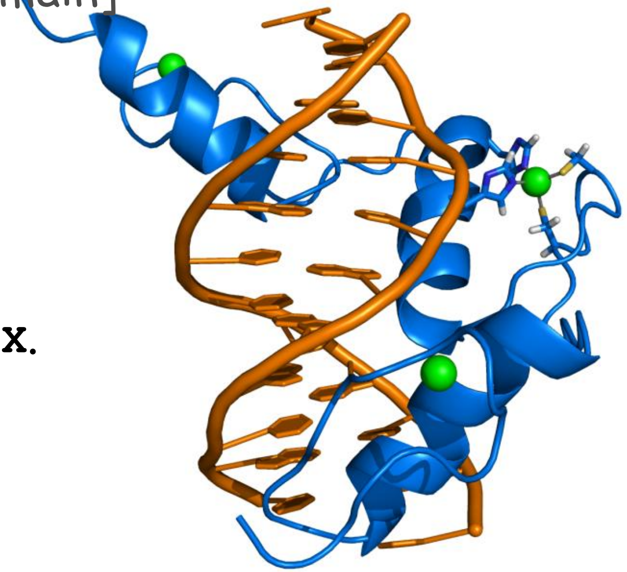
PROSITE. Huge database of protein families and domains.

Each **PROSITE** Pattern is a Protein "WORD" or sequence motif conserved in many sequences:

Q. How to describe a PROSITE pattern?

Ex. [signature of the C_2H_2-type zinc finger domain]

- C
- Between 2 and 4 amino acids.
- C
- 3 more amino acids.
- One of the following amino acids: **LIVMFYWCX**.
- 8 more amino acids.
- **H**
- Between 3 and 5 more amino acids.
- **H**



CAASCGGPYACGGWAGYHAGWH

Pattern Matching Applications

Test if a string matches some pattern.

- Process natural language.
- Scan for virus signatures.
- Access information in digital libraries.
- Search-and-replace in a word processors.
- Filter text (spam, NetNanny, ads, malware).
- Validate data-entry fields (dates, email, URL, credit card).
- Functionally annotate human proteome using PROSITE patterns.

Parse text files.

- Compile a Java program.
- Crawl and index the Web.
- Automatically create Java documentation from Javadoc comments.

Regular Expressions: Basic Operations

Regular expression. Notation to specify a set of strings.

operation	regular expression	matches	does not match
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 ababababa	aa abbba

Regular Expressions: Examples

Regular expression. Notation is surprisingly expressive.

regular expression	matches	does not match
`.*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

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	matches	does not match
one or more	**a(bc)+de**	abcde abcbcde	ade bcde
character class	**[A-Za-z][a-z]***	lowercase Capitalized	camelCase 4illegal
exactly k	**[0-9]{5}-[0-9]{4}**	08540-1321 19072-5541	111111111 166-54-1111
negation	**[^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];
      StdOut.println(input.matches(re));
   }
}
```

powerful string library method

C_2H_2 type zinc finger domain

```
% java Validate "C.{2,4}C...[LIVMFYWC].{8}H.{3,5}H" CAASCGGPYACGGAAGYHAGAH
true
```

legal Java identifier (simplified)

```
% java Validate "[A-Za-z][A-Za-z0-9]*" ident123
true
```

valid email address (simplified)

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

need quotes to "escape" the shell

String Searching Methods

`public class String`	*(Java's String library)*
`boolean matches(String re)`	*does this string match the given regular expression*
`String replaceAll(String re, String str)`	*replace all occurrences of regular expression with the replacement string*
`int indexOf(String r, int from)`	*return the index of the first occurrence of the string r after the index from*
`String[] split(String re)`	*split the string around matches of the given regular expression*

```
String s = StdIn.readAll();
s = s.replaceAll("\\s+", " ");
```

replace all sequences of whitespace characters with a single space

String Searching Methods

public class String	*(Java's String library)*
`boolean matches(String re)`	*does this string match the given regular expression*
`String replaceAll(String re, String str)`	*replace all occurrences of regular expression with the replacement string*
`int indexOf(String r, int from)`	*return the index of the first occurrence of the string r after the index from*
`String[] split(String re)`	*split the string around matches of the given regular expression*

```
String s = StdIn.readAll();
String[] words = s.split("\\s+");
```

create array of words in document

regular expression that
matches any whitespace character

DFAs

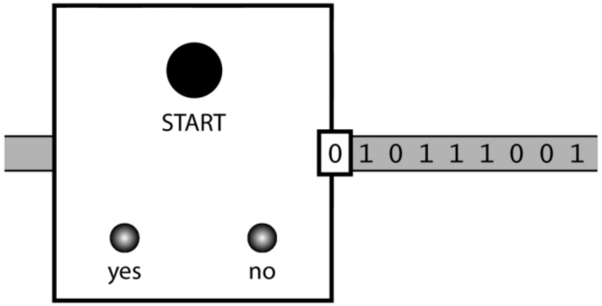
Solving the Pattern Match Problem

Regular expressions are a concise way to describe patterns.

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

DFA: simple machine that solves a pattern match problem.

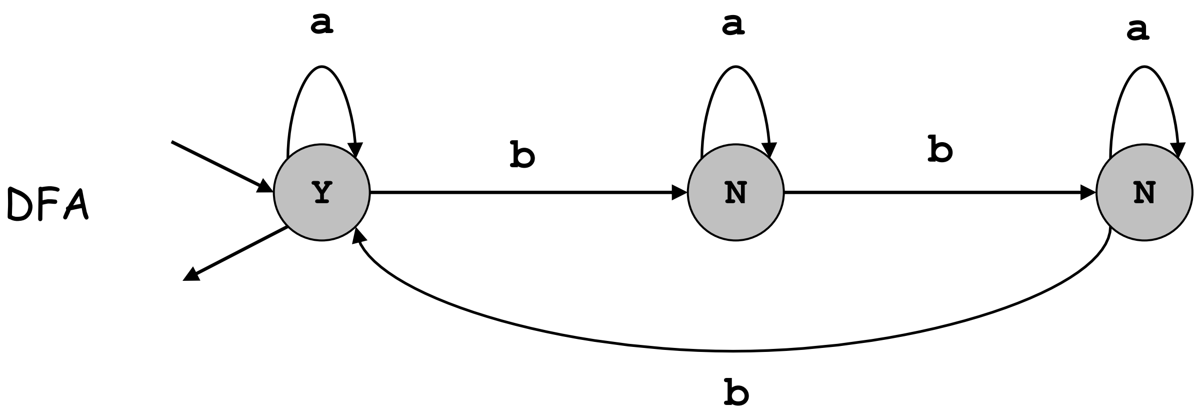
- Different machine for each pattern.
- Accepts or rejects string specified on input tape.
- Focus on `true` or `false` questions for simplicity.



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 input string if last state is labeled Y.



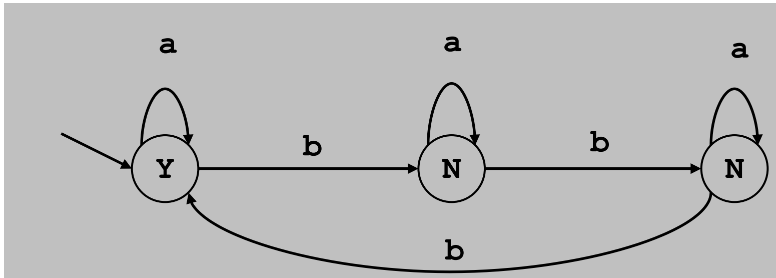
Input	b	b	a	a	b	b	a	b	b

DFA and RE Duality

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 RE that describes the same set of strings; for any RE, there exists a DFA that recognizes the same set.



multiple of 3 b's

```
a* | (a*ba*ba*ba*)*
```

multiple of 3 b's

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

Implementing a Pattern Matcher

Problem. Given a RE, 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.

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

Application: Harvester

Harvest information from input stream.

- Harvest patterns from DNA.

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

- Harvest email addresses from web for spam campaign.

```
% java Harvester "[a-z]+@([a-z]+\.)+(edu|com)" http://www.princeton.edu/~cos126
rs@cs.princeton.edu
maia@cs.princeton.edu
doug@cs.princeton.edu
wayne@cs.princeton.edu
```

Application: Harvester

Harvest information from input stream.

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

equivalent, but more efficient representation of a DFA

```
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()) {
         StdOut.println(matcher.group());
      }
   }
}
```

create NFA from RE

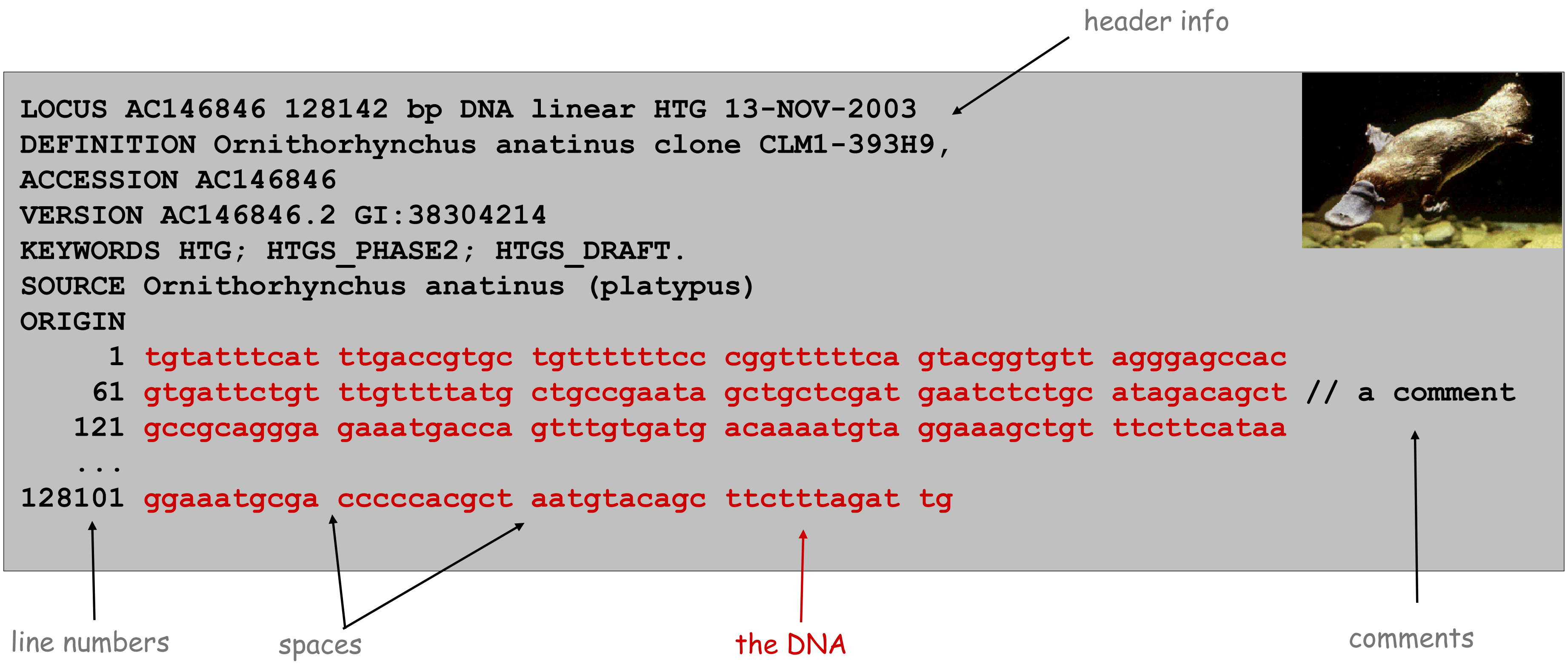
create NFA simulator

look for next match

the match most recently found

Application: Parsing a Data File

Ex: parsing an NCBI genome data file.



header info

```
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 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
```

line numbers

spaces

the DNA

comments

Application: Parsing a Data File

Ex: parsing an NCBI genome data file.

```
String re = "[ ]*[0-9]+([actg ]*).*";
Pattern pattern = Pattern.compile(re);
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
   }
}
```

extract the part of match in parens

```
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 tgtatttcat ttgaccgtgc tgtttttcc 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
```

Regular Expressions

WHENEVER I LEARN A NEW SKILL I CONCOCT ELABORATE FANTASY SCENARIOS WHERE IT LETS ME SAVE THE DAY.
OH NO! THE KILLER MUST HAVE FOLLOWED HER ON VACATION!
BUT TO FIND THEM WE'D HAVE TO SEARCH THROUGH 200 MB OF EMAILS LOOKING FOR SOMETHING FORMATTED LIKE AN ADDRESS!
IT'S HOPELESS!
EVERYBODY STAND BACK.
I KNOW REGULAR EXPRESSIONS.
PERL!
TAP TAP

http://xkcd.com/208/

Fundamental Questions

Q. Are there patterns that cannot be described by any RE/DFA?

A. Yes.

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

Q. Can we extend RE/DFA to describe richer patterns?

A. Yes.

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

Fundamental Questions

Q. Are there patterns that cannot be described by any RE/DFA?

A. Yes.

- Bit strings with equal number of 0s and 1s.

Proof Sketch

- Suppose that you have such a DFA, with N states.
- Give it N+1 0s followed by N+1 1s.
- Some state is revisited.
- Delete substring between visits.
- DFA recognizes that string, too.
- It does not have equal number of 0s and 1s.
- Contradiction!
- No such DFA exists.

Summary

Programmer.

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

Theoretician.

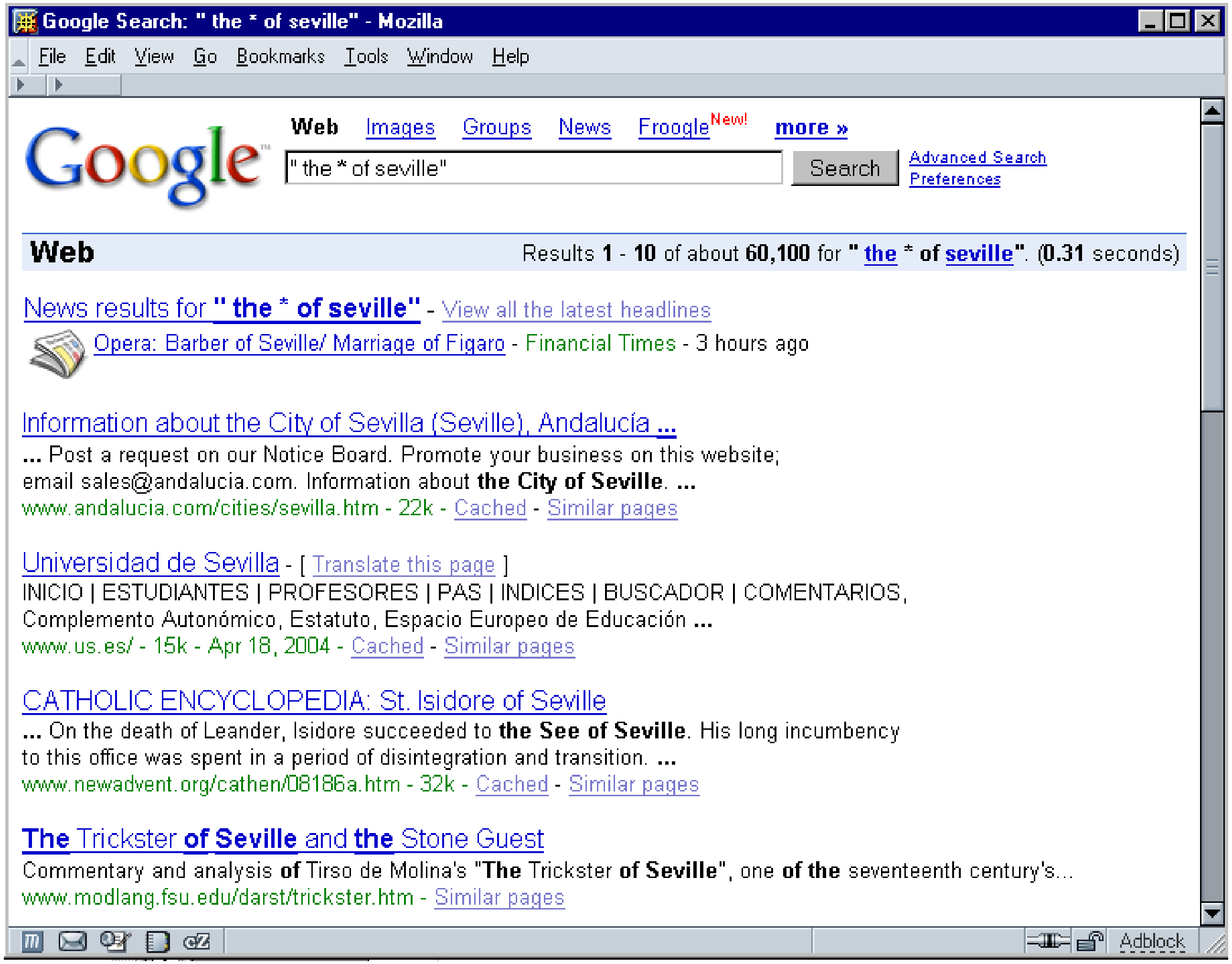
- RE is a compact description of a set of strings.
- DFA is an abstract machine that solves RE pattern match problem.

You. Practical application of core CS principles.

Extra Slides

Pattern Matching in Google

Google. Supports * for full word wildcard and | for union.



Google Search: " the * of seville" - Mozilla
File Edit View Go Bookmarks Tools Window Help
Web Images Groups News Froogle New! more »
Search
Advanced Search
Preferences
Adblock

" the * of seville"

Web Results **1** - **10** of about **60,100** for **" the * of seville"**. (**0.31** seconds)

News results for **" the * of seville"** - View all the latest headlines
Opera: Barber of Seville/ Marriage of Figaro - Financial Times - 3 hours ago

Information about the City of Sevilla (Seville), Andalucía ...
... Post a request on our Notice Board. Promote your business on this website; email sales@andalucia.com. Information about **the City of Seville**. ...
www.andalucia.com/cities/sevilla.htm - 22k - Cached - Similar pages

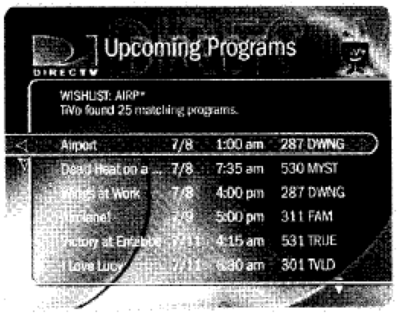
Universidad de Sevilla - [Translate this page]
INICIO | ESTUDIANTES | PROFESORES | PAS | INDICES | BUSCADOR | COMENTARIOS, Complemento Autonómico, Estatuto, Espacio Europeo de Educación ...
www.us.es/ - 15k - Apr 18, 2004 - Cached - Similar pages

CATHOLIC ENCYCLOPEDIA: St. Isidore of Seville
... On the death of Leander, Isidore succeeded to **the See of Seville**. His long incumbency to this office was spent in a period of disintegration and transition. ...
www.newadvent.org/cathen/08186a.htm - 32k - Cached - Similar pages

The Trickster **of Seville** and **the** Stone Guest
Commentary and analysis **of** Tirso de Molina's "**The** Trickster **of Seville**", one **of the** seventeenth century's...
www.modlang.fsu.edu/darst/trickster.htm - Similar pages

Pattern Matching in TiVo

TiVo. WishList has very limited pattern matching.



Upcoming Programs

WISHLIST: AIRP*
TiVo found 25 matching programs.

Airport	7/8	1:00 am	287 DWNG
Dead Heat on a ...	7/8	7:35 am	530 MYST
Wings at Work	7/8	4:00 pm	287 DWNG
Airplane!	7/9	5:00 pm	311 FAM
Victory at Entebbe	7/11	4:15 am	531 TRUE
I Love Lucy	7/11	6:30 am	301 TVLD

Using * in WishList Searches. To search for similar words in Keyword and Title WishList searches, use the asterisk (*) as a special symbol that replaces the endings of words. For example, the keyword *AIRP** would find shows containing "airport," "airplane," "airplanes," as well as the movie "Airplane!" To enter an asterisk, press the SLOW (⏵) button as you are spelling out your keyword or title.

The asterisk can be helpful when you're looking for a range of similar words, as in the example above, or if you're just not sure how something is spelled. Pop quiz: is it "irresistible" or "irresistable?" Use the keyword *IRRESIST** and don't worry about it! Two things to note about using the asterisk:

- It can only be used at a word's end; it cannot be used to omit letters at the beginning or in the middle of a word. (For example, *AIR*NE* or **PLANE* would not work.)

Reference: page 76, Hughes DirectTV TiVo manual