**Introduction to Theoretical CS**

**Fundamental questions:**
- Q. What can a computer do?
- Q. 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.

**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**
Pattern Matching

Pattern matching problem. Is a given string in a specified set of strings?

Ex. [genomics]
• Fragile X syndrome is a common cause of mental retardation.
• Human genome contains triplet repeats of coa or xaa,
  bracketed by aca at the beginning and cta at the end.
• Number of repeats is variable, and correlated with syndrome.

First step:

Regular expression. A formal notation for specifying a set of strings.

Specified set of strings: "all strings of G, C, T, A having some occurrence
of CGG followed by any number of CGG or AGG triplets, followed by CTG"
Q: "Is this string in the set?"
A: Yes "GCGCGGAGGCGGCTG"

PROSITE. Huge database of protein families and domains.

Q. How to describe a protein motif?

Ex. [signature of the C_{2}H_{2}-type zinc finger domain]
1. C
2. Between 2 and 4 amino acids.
3. C
4. 3 more amino acids.
5. One of the following amino acids: LIVMFYWX.
6. 8 more amino acids.
7. H
8. Between 3 and 5 more amino acids.
9. H

A. Use a regular expression.

Pattern Matching Application

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, 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.

Regular Expressions: Basic Operations

Regular expression. Notation to specify a set of strings.

<table>
<thead>
<tr>
<th>operation</th>
<th>regular expression</th>
<th>matches</th>
<th>does not match</th>
</tr>
</thead>
<tbody>
<tr>
<td>concatenation</td>
<td>aabaab</td>
<td>aabaab</td>
<td>every other string</td>
</tr>
<tr>
<td>wildcard</td>
<td>.u.u.u.</td>
<td>cumulus jugulum</td>
<td>succubus tumultuous</td>
</tr>
<tr>
<td>union</td>
<td>aa</td>
<td>baab</td>
<td>aa baab</td>
</tr>
<tr>
<td>closure</td>
<td>ab*a</td>
<td>aa abba</td>
<td>ab ababa</td>
</tr>
<tr>
<td>parentheses</td>
<td>a(ab)b</td>
<td>aab</td>
<td>aab abab</td>
</tr>
<tr>
<td></td>
<td>(ab)*a</td>
<td>a ababababa</td>
<td>aa abba</td>
</tr>
</tbody>
</table>
Regular Expressions: Examples

**Regular expression.** 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>`a*</td>
<td>(a<em>ba</em>ba<em>ba</em>)*`</td>
<td>bbb aaa bbaaabbbaa</td>
</tr>
<tr>
<td><code>.*0 . . . </code></td>
<td>fifth to last digit is 0</td>
<td>1000234 98701234</td>
</tr>
<tr>
<td>gcgcgg</td>
<td>gcgcggctg ggcgcgaggctg</td>
<td>gcgcggctg gcgcggaggctg</td>
</tr>
</tbody>
</table>

Regular Expression Challenge 1

**Q.** Consider the RE

\[ a*bb(ab|ba)^* \]

Which of the following strings match (is in the set it describes)?

a. abb  
b. abba  
c. aaba  
d. bbbab  
e. cbb  
f. bbbabbbab

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 1: \([a-e]^{+}\) is shorthand for \((a|b|c|d|e)(a|b|c|d|e)^*\).  
  - Ex 2: \(\text{s}\) is shorthand for “any whitespace character” (space, tab, ...).

<table>
<thead>
<tr>
<th>operation</th>
<th>regular expression</th>
<th>matches</th>
<th>does not match</th>
</tr>
</thead>
<tbody>
<tr>
<td>one or more</td>
<td>a(bc)+de</td>
<td>abcde</td>
<td>ade bde</td>
</tr>
<tr>
<td>character class</td>
<td>[A-Za-z]+</td>
<td>lowercase, capitalized</td>
<td>camelCase, illegal</td>
</tr>
<tr>
<td>exactly k</td>
<td>[0-9]{5}-[0-9]{4}</td>
<td>08540-1321, 19072-5541</td>
<td>11111111, 166-54-1111</td>
</tr>
<tr>
<td>negation</td>
<td>[^aeiou][6]</td>
<td>rhythm</td>
<td>decade</td>
</tr>
</tbody>
</table>

Regular Expression Challenge 2

**Q.** Give an RE that describes the following set of strings:

- characters are \(A, C, T\) or \(G\)  
- starts with \(ATG\)  
- length is a multiple of 3  
- ends with \(TAG, TAA,\) or \(TTG\)
Describing a Pattern

PROSITE. Huge database of protein families and domains.

Q. How to describe a protein motif?

Ex. [signature of the C2H2-type zinc finger domain]
1. C
2. Between 2 and 4 amino acids.
3. C
4. 3 more amino acids.
5. One of the following amino acids: LIVMFYWC.
6. 8 more amino acids.
7. H
8. Between 3 and 5 more amino acids.
9. H

A. C.{2,4}C...[LIVMFYWC].{8}H.{3,5}H

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

REs in Java

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

REs in Java

```java
public class Validate {
    public static void main(String[] args) {
        String re = args[0];
        String input = args[1];
        StdOut.println(input.matches(re));
    }
}
```
REs in Java

```java
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 an array of the words in StdIn
}
```

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.
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.

Practical consequence of duality proof: to match RE
- build DFA
- simulate DFA on input string.

Q. Consider this DFA:

Which of the following sets of strings does it recognize?

a. Bitstrings with at least one 1
b. Bitstrings with an equal number of occurrences of 01 and 10
c. Bitstrings with more 1s than 0s
d. Bitstrings with an equal number of occurrences of 0 and 1
e. Bitstrings that end in 1

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.

```java
State state = start;
while (!StdIn.isEmpty())
{
    char c = StdIn.readChar();
    state = state.next(c);
}
StdOut.println(state.accept());
```
Harvest information from input stream.

- Harvest patterns from DNA.
  ```java
  % java Harvester "gcg(cgg|agg)*ctg" chromosomeX.txt
  gcgcggcggcggcggcggctg
gcgctg
gcgctg
ggcgcggcggcggaggcggaggcggctg
  
  % java Harvester "gcg(cgg|agg)*ctg" chromosomeX.txt
  gcgcggcggcggcggcggctg
gcgctg
gcgctg
ggcgcggcggcggaggcggaggcggctg
  ```

- Harvest email addresses from web for spam campaign.
  ```java
  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
  ```

Real-World Application: Parsing a Data File

Java's Pattern and Matcher classes
- use REs for pattern matching (previous slide)
- extend REs to facilitate processing string-based data

Ex: parsing an NCBI genome data file.

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;
public class ParseNCBI {
    public static void main(String[] args)
    {
        String re = \[0-9]+\((actg )*\)+;\.
        Pattern pattern = Pattern.compile(re);
        In in = new In(args[0]);
        String data = ""
        while (!in.isEmpty())
        {
            String line = in.readLine();
            Matcher matcher = pattern.matcher(line);
            if (matcher.find())
                data += matcher.group(1).replaceAll(" ", "");
        }
        System.out.println(data);
    }
}
```

Goal. Extract the data as a single actg string.

```
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
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

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
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
```
Real-World Application: Parsing a Data File

```java
import java.util.regex.Pattern;
import java.util.regex.Matcher;
public class ParseNCBI {
    public static void main(String[] args) {
        String re = "\[ \]*[0-9]+(\[actg \]*).*";
        Pattern pattern = Pattern.compile(re);
        In in = new In(args[0]);
        String data = "";
        while (!in.isEmpty()) {
            String line = in.readLine();
            Matcher matcher = pattern.matcher(line);
            if (matcher.find())
                data += matcher.group(1).replaceAll(" ", "");
        }
        System.out.println(data);
    }
}
```

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.

**Variations**
- Yes (accept) and No (reject) states sometimes drawn differently
- Terminology: Deterministic Finite State Automaton (DFA), Finite State Machine (FSM), Finite State Automaton (FSA) are the same
- DFA's can have output, specified on the arcs or in the states
  - These may not have explicit Yes and No states

**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.
- and many, many more . . .

**Q.** Can we extend RE/DFA to describe richer patterns?
**A.** Yes.
- Context free grammar (e.g., Java).
- Turing machines.

Limitations of DFA

No DFA can recognize the language of all bit strings with an equal number of 0’s and 1’s.
- Suppose some N-state DFA can recognize this language.
- Consider following input: 0000000011111111
  - Our DFA must accept this string.
  - Some state x is revisited during first N+1 0’s since only N states.
  - Machine would accept same string without intervening 0’s.

- This string doesn’t have an equal number of 0’s and 1’s.
7.4 Turing Machines

Desiderata. Simple model of computation that is "as powerful" as conventional computers.

Intuition. Simulate how humans calculate.

Ex. Addition.

Tape.
- Stores input, output, and intermediate results.
- One arbitrarily long strip, divided into cells.
- Finite alphabet of symbols.

Tape head.
- Points to one cell of tape.
- Reads a symbol from active cell.
- Writes a symbol to active cell.
- Moves left or right one cell at a time.

States.
- Finite number of possible machine configurations.
- Determines what machine does and which way tape head moves.

State transition diagram.
- Ex. if in state 2 and input symbol is 1 then: overwrite the 1 with x, move to state 0, move tape head to left.
Turing Machine: Fetch, Execute

**States.**
- Finite number of possible machine configurations.
- Determines what machine does and which way tape head moves.

**State transition diagram.**
- Ex. if in state 2 and input symbol is 1 then: overwrite the 1 with x, move to state 0, move tape head to left.

---

Turing Machine: Initialization and Termination

**Initialization.**
- Set input on some portion of tape.
- Set tape head.
- Set initial state.

---

**Termination.**
- Stop if enter yes, no, or halt state.
- Infinite loop possible.
  -(definitely stay tuned !)

---

Turing Machine Summary

**Goal:** simplest machine that is "as powerful" as conventional computers.

**Surprising Fact 1.** Such machines are very simple: TM is enough!
**Surprising Fact 2.** Some problems cannot be solved by ANY computer.

**Consequences.**
- Precursor to general purpose programmable machines.
- Exposes fundamental limitations of all computers.
- Enables us to study the physics and universality of computation.
- No need to seek more powerful machines!

**Variations**
- Instead of just recognizing strings, TM's can produce output: the contents of the tape.
- Instead of Y and N states, TM's can have a plain Halt state.
Alan Turing (1912-1954).

- Father of computer science.
- Computer Science's "Nobel Prize" is called the Turing Award.

Alan's report card at 14.

Alan Turing and his elder brother.