MA/CSSE 474

Theory of Computation

Course Intro

Introductions

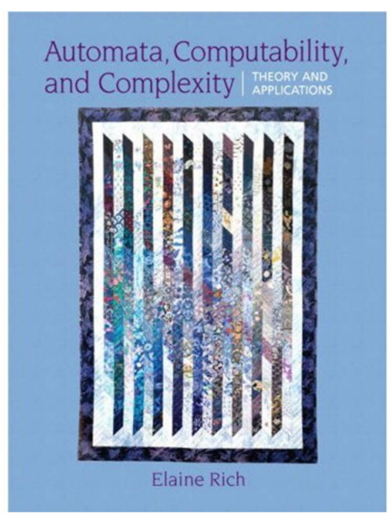
- Students
  - Normally I would have everyone introduce themselves in class, but this time the class is so big that I am having you introduce yourself on an ANGEL discussion forum
  - Introduce yourself, respond to someone else's intro. Before tomorrow's class. Mine is already there.
- Graders: Kurtis Zimmerman, Kenny Gao, Eric Reed
- Instructor: Claude Anderson: F-210, x8331

Instructor Professional Background

- **Formal Education:**
  - BS Caltech, Mathematics 1975
  - Ph.D. Illinois, Mathematics 1981
  - MS Indiana, Computer Science 1987
- **Teaching:**
  - TA at Illinois, Indiana 1975-1981, 1986-87
  - Wilkes College (now Wilkes University) 1981-88
  - RHIT 1988 – 2022**?**
- **Major Consulting Gigs:**
  - Pennsylvania Funeral Directors Assn 1983-88
  - Navistar International 1994-95
  - Beckman Coulter 1996-98
  - ANGEL Learning 2005-2008
- **Theory of Computation history**

Textbook

- Fairly new
- Thorough
- Literate
- Large
- Theory and Applications
- We'll focus more on theory; applications there for you to see



Online Materials Quick Tour

- On the Web – general stuff
  - Suggestion: bookmark schedule page
- On ANGEL – personal stuff
  - surveys, solutions, discussions, grades
  - Suggestion: subscribe to discussion forums.
- Many things are under construction and subject to change, especially the course schedule.
- HW due Friday day usually posted by Tuesday
- HW due Tuesday usually posted by Friday
- Preliminary versions already there.

Grateful Acknowledgement

- Many of the PowerPoint slides that I will use were produced by Elaine Rich, in conjunction with her textbook.
- I will modify some of them.
- I will create some new ones.

Why Study the Theory of Computation?

- Why not just write programs?
- Implementations come and go.

IBM 7090 Programming in the 1950's

```
ENTRY      SXA     4,RETURN
           LDQ     X
           FMP     A
           FAD     B
           XCA
           FMP     X
           FAD     C
           STO     RESULT
RETURN     TRA     0
A          BSS     1
B          BSS     1
C          BSS     1
X          BSS     1
TEMP       BSS     1
STORE      BSS     1
           END
```

Programming in the 1970's (IBM 360)

```
//MYJOB    JOB (COMPRESS),
          'VOLKER BANDKE',CLASS=P,COND=(0,NE)
//BACKUP  EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  DISP=SHR,DSN=MY.IMPORTNT.PDS
//SYSUT2   DD  DISP=(,CATLG),
           DSN=MY.IMPORTNT.PDS.BACKUP,
//         UNIT=3350,VOL=SER=DISK01,
//         DCB=MY.IMPORTNT.PDS,
           SPACE=(CYL,(10,10,20))
//COMPRESS EXEC PGM=IEBCOPY
//SYSPRINT DD  SYSOUT=*
//MYPDS    DD  DISP=OLD,DSN=*.BACKUP.SYSUT1
//SYSIN    DD  *
COPY INDD=MYPDS,OUTDD=MYPDS
//DELETE2 EXEC PGM=IEFBR14
//BACKPDS  DD  DISP=(OLD,DELETE,DELETE),
          DSN=MY.IMPORTNT.PDS.BACKUP
```

Programming in the New Millennium

```
public static TreeMap<String, Integer> create() throws IOException {
    Integer freq;
    String word;
    TreeMap< String, Integer> result = new TreeMap<String, Integer>();
    JFileChooser c = new JFileChooser();
    int retval = c.showOpenDialog(null);
    if (retval == JFileChooser.APPROVE_ OPTION) {
        Scanner s = new Scanner(c.getSelectedFile());
        while(s.hasNext()) {
            word = s.next().toLowerCase();
            freq = result.get(word);
            result.put(word, (freq == null ? 1 : freq + 1));
        }
    }
    return result;
}
```

Find the frequency of each word in a text file chosen by the user.

Timeless Abstractions

- Mathematical properties of problems and algorithms.
- Do not depend on technology or programming style.
- The basic principles remain the same:
- What can be computed, and what cannot?
- What are reasonable mathematical models of computation?

Q1

More Detailed Questions

- Does a computational solution to the problem exist?
  - Without regard to limitations of processor speed or memory size.
  - If not, is there a restricted but useful variation of the problem for which a solution does exist?
- If a solution exists, can it be implemented using some fixed amount of memory?
- If a solution exists, how efficient is it?
  - More specifically, how do its time and space requirements grow as the size of the problem grows?
- Are there groups of problems that are equivalent in the sense that if there is an efficient solution to one member of the group there is an efficient solution to all the others?

Applications of the Theory

- Finite State Machines (FSMs) for parity checkers, vending machines, communication protocols, and building security devices.
- Interactive games as nondeterministic FSMs.
- Programming languages, compilers, and context-free grammars.
- Natural languages are mostly context-free. Speech understanding systems use probabilistic FSMs.
- Computational biology: DNA and proteins are strings.
- The undecidability of a simple security model.
- Artificial intelligence: the undecidability of first-order logic.

What we will focus on

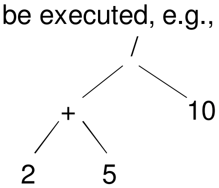
- Definitions
- Theorems
- Examples
- Proofs

Languages and Strings

Some Language-related Problems

```
int alpha, beta;
alpha = 3;
beta = (2 + 5) / 10;
```

(1) **Lexical analysis**: Scan the program and break it up into variable names, numbers, operators, punctiuation, etc.
(2) **Parsing**: Create a tree that corresponds to the sequence of operations that should be executed, e.g.,



(3) **Optimization**: Realize that we can skip the first assignment since the value is never used and that we can pre-compute the arithmetic expression, since it contains only constants.
(4) **Termination**: Decide whether the program is guaranteed to halt.
(5) **Interpretation**: Figure out what (if anything) useful it does.

A Framework for Analyzing Problems

We need a single framework in which we can analyze a very diverse set of problems.

The framework we will use is

Language Recognition

A (formal) ***language*** is a (possibly infinite) set of finite-length strings over a finite alphabet.

Q2

Strings

A ***string*** is a finite sequence, possibly empty, of symbols from some finite alphabet Σ.

- ε is the empty string (some books/papers use λ instead)
- Σ* is the set of all possible strings over an alphabet Σ

Alphabet name	***Alphabet symbols***	***Example strings***
The English alphabet	`{a, b, c, …, z}`	ε, `aabbcg`, `aaaaa`
The binary alphabet	`{0, 1}`	ε, `0`, `001100`
A star alphabet	{★ , ✪ , ☆ , ✯, ✡, ✰}	ε, ✪✪, ✪✯✯✰★✰
A music alphabet	{𝅝, 𝅗𝅥, ♩., ♪, 𝅘𝅥𝅯, 𝅘𝅥𝅰, ●}	ε, 𝅝𝅗𝅥𝅗𝅥𝅗𝅥♩♩♩

Q3

Functions on Strings

Counting: $|s|$ is the number of symbols in s.

$|\varepsilon| = 0$
$|\texttt{1001101}| = 7$

$\#_c(s)$ is the number of times that c occurs in s.

$\#_{\texttt{a}}(\texttt{abbaaa}) = 4$.

More Functions on Strings

Concatenation: st is the ***concatenation*** of s and t.

If x = `good` and y = `bye`, then xy = `goodbye`.

Note that $|xy| = |x| + |y|$.

ε is the **identity** for concatenation of strings. So:

$\forall x\ (x\,\varepsilon = \varepsilon\, x = x)$.

Concatenation is **associative**. So:

$\forall s, t, w\ ((st)w = s(tw))$.

More Functions on Strings

Replication: For each string w and each natural number i, the string w^i is:

$w^0 = \varepsilon$
$w^{i+1} = w^i\ w$

Examples:

$\texttt{a}^3 = \texttt{aaa}$
$(\texttt{bye})^2 = \texttt{byebye}$
$\texttt{a}^0\texttt{b}^3 = \texttt{bbb}$

More Functions on Strings

Reverse: For each string w, w^R is defined as:

if $|w| = 0$ then $w^R = w = \varepsilon$

if $|w| \geq 1$ then:
$\exists a \in \Sigma\ (\exists u \in \Sigma^*\ (w = ua))$.
So define $w^R = a\ u^R$.

Concatenation and Reverse of Strings

Theorem: If w and x are strings, then $(w\,x)^R = x^R\,w^R$.

Example:

$(\texttt{nametag})^R = (\texttt{tag})^R\,(\texttt{name})^R = \texttt{gateman}$

Q4

Concatenation and Reverse of Strings

Proof: By induction on $|x|$: This slide is hidden. Do it on the board.

$|x| = 0$: Then $x = \varepsilon$, and $(wx)^R = (w\,\varepsilon)^R = (w)^R = \varepsilon\,w^R = \varepsilon^R\,w^R = x^R\,w^R$.

$\forall n \geq 0\ (((|x| = n) \rightarrow ((w\,x)^R = x^R\,w^R)) \rightarrow$
$((|x| = n + 1) \rightarrow ((w\,x)^R = x^R\,w^R)))$:

Consider any string x, where $|x| = n + 1$. Then $x = u\,a$ for some character a and $|u| = n$. So:

$(w\,x)^R$	$= (w\,(u\,a))^R$	rewrite x as ua
	$= ((w\,u)\,a)^R$	associativity of concatenation
	$= a\,(w\,u)^R$	definition of reversal
	$= a\,(u^R\,w^R)$	induction hypothesis
	$= (a\,u^R)\,w^R$	associativity of concatenation
	$= (ua)^R\,w^R$	definition of reversal
	$= x^R\,w^R$	rewrite ua as x

Relations on Strings: Substring

`aaa`	is a ***substring*** of	`aaabbbaaa`
`aaaaaa`	is not a substring of	`aaabbbaaa`
`aaa`	is a ***proper substring*** of	`aaabbbaaa`

Every string is a substring of itself.

ε is a substring of every string.

Relations on Strings: Prefix

s is a ***prefix*** of t iff: $\exists x \in \Sigma^* \, (t = sx)$.

s is a ***proper prefix*** of t iff: s is a prefix of t and $s \neq t$.

Examples:

The ***prefixes*** of `abba` are: ε, `a`, `ab`, `abb`, `abba`.
The ***proper prefixes*** of `abba` are: ε, `a`, `ab`, `abb`.

Every string is a prefix of itself.

ε is a prefix of every string.

Relations on Strings: Suffix

s is a ***suffix*** of t iff: $\exists x \in \Sigma^*$ ($t = xs$).

s is a ***proper suffix*** of t iff: s is a suffix of t and $s \neq t$.

Examples:

The ***suffixes*** of `abba` are: ε, `a`, `ba`, `bba`, `abba`.
The ***proper suffixes*** of `abba` are: ε, `a`, `ba`, `bba`.

Every string is a suffix of itself.

ε is a suffix of every string.

Defining a Language

A ***language*** is a (finite or infinite) set of strings over a finite alphabet Σ.

Examples: Let Σ = {`a`, `b`}

Some languages over Σ:
∅,
{ε},
{`a`, `b`},
{ε, `a`, `aa`, `aaa`, `aaaa`, `aaaaa`}

The language Σ^* contains an infinite number of strings, including: ε, `a`, `b`, `ab`, `ababaa`.

Example Language Definitions

$L = \{x \in \{\mathtt{a}, \mathtt{b}\}^* : \text{all } \mathtt{a}\text{'s precede all } \mathtt{b}\text{'s}\}$

ε, a, aa, aabbb, and bb are in *L*.

aba, ba, and abc are not in *L*.

Example Language Definitions

$L = \{x : \exists y \in \{\mathtt{a}, \mathtt{b}\}^* : x = y\mathtt{a}\}$

Simple English description:

The Perils of Using English

$L = \{x\#y$: x, $y \in$ {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}* and, when x and y are viewed as the decimal representations of natural numbers, *square*(x) = y}.

Examples:

`3#9, 12#144`

`3#8, 12, 12#12#12`

`#`

More Example Language Definitions

$L = \{\ \} = \varnothing$

$L = \{\varepsilon\}$

English

$L = \{w$: w is a sentence in English$\}$.

Examples:

Kerry hit the ball.

Colorless green ideas sleep furiously.

The window needs fixed.

Ball the Stacy hit blue.

A Halting Problem Language

$L = \{w$: w is a C program that halts on all inputs$\}$.

- Well specified.
- Can we decide what strings it contains?

Languages and Prefixes

What are the following languages?

$L = \{w \in \{\texttt{a}, \texttt{b}\}^*$: no prefix of w contains $\texttt{b}\}$

$L = \{w \in \{\texttt{a}, \texttt{b}\}^*$: no prefix of w starts with $\texttt{a}\}$

$L = \{w \in \{\texttt{a}, \texttt{b}\}^*$: every prefix of w starts with $\texttt{a}\}$

Q5

Using Replication in a Language Definition

$L = \{\texttt{a}^n : n \geq 0\}$

Q6-7