Welcome to
CS 620 –
Theory of Computation

Fall 2016
Instructor: Marc Pomplun

Instructor – Marc Pomplun
Office: S-3-171
Lab: S-3-135
Office Hours: Tuesdays and Thursdays 3:15 – 4:00pm and 5:15 – 6:00pm and/or by appointment
Phone: 617-287-6443
E-Mail: marc@cs.umb.edu
Homepage: www.cs.umb.edu/~marc/cs620/

I’m from Lübeck, Germany:

My Research: Visual Attention in Humans and Machines

Example: Distribution of Visual Attention

Selectivity in Complex Scenes
Modeling of Brain Functions

- unit and connection in the interpretive network
- unit and connection in the gating network
- unit and connection in the top-down bias network

Computer Vision:

Human-Computer Interfaces:

Now back to CS 620:

Textbook:

On the Web:
http://www.cs.umb.edu/~marc/cs620/ (contains all kinds of course information and also my slides in PPTX and PDF formats, updated after each session)

Mailing List
Please use the ‘apply’ command on the UNIX system to register for our mailing list (CS620, Section 1). I expect everyone to be on the list, because I will use it to make announcements.

Also, I would like to encourage you to use the list for discussion. If you have a question that you think is important for many students in the course, please send it to the list at cs620-1@cs.umb.edu and I will respond to the list.

Send all other questions to me. For these questions, I will send my reply only to you.

Your Evaluation

- 6 sets of exercises 18%
- midterm (1.5 hours) 37%
- final exam (2.5 hours) 45%
Grading
For the assignments, exams and your course grade, the following scheme will be used to convert percentages into letter grades:

- \[ \geq 95\%: \text{A} \]
- \[ \geq 90\%: \text{A-} \]
- \[ \geq 74\%: \text{C+} \]
- \[ \geq 70\%: \text{C} \]
- \[ \geq 66\%: \text{C-} \]
- \[ \geq 86\%: \text{B+} \]
- \[ \geq 82\%: \text{B} \]
- \[ \geq 78\%: \text{B-} \]
- \[ \geq 62\%: \text{D+} \]
- \[ \geq 56\%: \text{D} \]
- \[ \geq 50\%: \text{D-} \]
- < 50\%: F

Complaints about Grading
If you think that the grading of your assignment or exam was unfair,
- write down your complaint (handwriting is OK),
- attach it to the assignment or exam,
- and give it to me or put it in my mailbox.
I will re-grade the whole exam/assignment and return it to you in class.

What is so interesting about the Theory of Computation?
- Theory of Computation is the most fundamental subject in computer science.
- What you learn in this course applies to all computers and all programming languages that will ever exist.
- You will understand the capabilities of algorithms in general.
- For example, you will learn about problems that cannot be solved algorithmically.

Novelty: The Haskell Programming Language
I will give a brief introduction to Haskell as a tool for studying the theory of computation. We will not have enough time for a thorough introduction, but if you are interested, you can study for yourself:
http://learnyouahaskell.com/
http://book.realworldhaskell.org/
I recommend that you read Chapters 1 and 2 of "Learn you a Haskell" and experiment with the language a bit.

Example: Quicksort in Haskell:
qsort [] = []
qsort (x:xs) = qsort [y | y < xs, y < x] ++ [x] ++ qsort [y | y < xs, y >= x]

Novelty: The Haskell Programming Language
You are not required to learn Haskell, and no exams or assignments will ask any questions about it. You can use it to "play around" with the concepts introduced in class.
I will provide Haskell code for various subjects.
Please download the Haskell Platform here:
http://www.haskell.org
Preliminaries

- Sets and n-tuples
- Functions
- Alphabets and Strings
- Predicates
- Quantifiers
- Proof by Contradiction
- Mathematical Induction

Set Theory

- Set: Collection of objects ("elements")
- \( a \in A \)  "a is an element of A"
- \( a \notin A \)  "a is not an element of A"
- \( A = \{a_1, a_2, \ldots, a_n\} \)  "A contains..."
- Order of elements is meaningless
- It does not matter how often the same element is listed.

Cartesian Product

The ordered n-tuple \((a_1, a_2, a_3, \ldots, a_n)\) is an ordered collection of objects.

Two ordered n-tuples \((a_1, a_2, a_3, \ldots, a_n)\) and \((b_1, b_2, b_3, \ldots, b_n)\) are equal if and only if they contain exactly the same elements in the same order, i.e. \( a_i = b_i \) for \( 1 \leq i \leq n \).

The Cartesian product of two sets is defined as:
\[ A \times B = \{(a, b) | a \in A \land b \in B\} \]

Example:
\( A = \{x, y\} \), \( B = \{a, b, c\} \)

\[ A \times B = \{(x, a), (x, b), (x, c), (y, a), (y, b), (y, c)\} \]

The Cartesian product of two or more sets is defined as:
\[ A_1 \times A_2 \times \ldots \times A_n = \{(a_1, a_2, \ldots, a_n) | a_i \in A_i \text{ for } 1 \leq i \leq n\} \]

Set Operations

Union: \( A \cup B = \{x | x \in A \lor x \in B\} \)
Example: \( A = \{a, b\} \), \( B = \{b, c, d\} \)
\[ A \cup B = \{a, b, c, d\} \]

Intersection: \( A \cap B = \{x | x \in A \land x \in B\} \)
Example: \( A = \{a, b\} \), \( B = \{b, c, d\} \)
\[ A \cap B = \{b\} \]

Set Operations

Two sets are called disjoint if their intersection is empty, that is, they share no elements:
\( A \cap B = \emptyset \)

The difference between two sets \( A \) and \( B \) contains exactly those elements of \( A \) that are not in \( B \):
\( A - B = \{x | x \in A \land x \notin B\} \)

Example: \( A = \{a, b\} \), \( B = \{b, c, d\} \)
\[ A - B = \{a\} \]
Set Operations

The complement of a set $A$ contains exactly those elements under consideration that are not in $A$: $-A = U - A$

Example: $U = \mathbb{N}$, $B = \{250, 251, 252, \ldots\}$
$-B = \{0, 1, 2, \ldots, 248, 249\}$

Functions

A function $f$ from a set $A$ to a set $B$ is an assignment of exactly one element of $B$ to each element of $A$.

We write $f(a) = b$
if $b$ is the unique element of $B$ assigned by the function $f$ to the element $a$ of $A$.

If $f$ is a function from $A$ to $B$, we write $f: A \to B$
(note: Here, "$\to$" has nothing to do with if… then)

Functions

If $f: A \to B$, we say that $A$ is the domain of $f$ and $B$ is the codomain of $f$.

If $f(a) = b$, we say that $b$ is the image of $a$ and $a$ is the pre-image of $b$.

The range of $f: A \to B$ is the set of all images of elements of $A$.

We say that $f: A \to B$ maps $A$ to $B$.

Alphabets and Strings

An alphabet is a finite, nonempty set $A$ of objects called symbols.

A word (string) on $A$ is an $n$-tuple of symbols of $A$.
Instead of using the notation $(a_1, a_2, \ldots, a_n)$ we can just write $a_1a_2\ldots a_n$.
The set of all words on $A$ is written $A^*$.
Any subset of $A^*$ is called a language on $A$.

Propositional Functions

Propositional function (open sentence): statement involving one or more variables, e.g.: $x - 3 > 5$.
Let us call this propositional function $P(x)$, where $P$ is the predicate and $x$ is the variable.

What is the truth value of $P(2)$? false
What is the truth value of $P(8)$? false
What is the truth value of $P(9)$? true
Propositional Functions

Let us consider the propositional function $Q(x, y, z)$ defined as:

$x + y = z$.

Here, $Q$ is the predicate and $x$, $y$, and $z$ are the variables.

What is the truth value of $Q(2, 3, 5)$? true
What is the truth value of $Q(0, 1, 2)$? false
What is the truth value of $Q(9, -9, 0)$? true

Universal Quantification

Let $P(x)$ be a propositional function.

**Universally quantified sentence:**

For all $x$ in the universe of discourse $P(x)$ is true.

Using the universal quantifier $\forall$:

$\forall x P(x)$ “for all x $P(x)$” or “for every x $P(x)$”

(Note: $\forall x P(x)$ is either true or false, so it is a proposition, not a propositional function.)

Universal Quantification

Example:

$S(x)$: $x$ is a UMB student.
$G(x)$: $x$ is a genius.

What does $\forall x (S(x) \rightarrow G(x))$ mean?

“If $x$ is a UMB student, then $x$ is a genius.”
or

“All UMB students are geniuses.”

Existential Quantification

**Existentially quantified sentence:**

There exists an $x$ in the universe of discourse for which $P(x)$ is true.

Using the existential quantifier $\exists$:

$\exists x P(x)$ “There is an $x$ such that $P(x)$.”

(“There is at least one $x$ such that $P(x)$.”

(Note: $\exists x P(x)$ is either true or false, so it is a proposition, but no propositional function.)

Existential Quantification

Example:

$P(x)$: $x$ is a UMB professor.
$G(x)$: $x$ is a genius.

What does $\exists x (P(x) \land G(x))$ mean?

“There is an $x$ such that $x$ is a UMB professor and $x$ is a genius.”
or

“At least one UMB professor is a genius.”

Disproof by Counterexample

A counterexample to $\forall x P(x)$ is an object $c$ so that $P(c)$ is false.

Statements such as $\forall x (P(x) \rightarrow Q(x))$ can be disproved by simply providing a counterexample.

Statement: “All birds can fly.”

Disproved by counterexample: Penguin.
The principle of mathematical induction is a useful tool for proving that a certain predicate is true for all natural numbers.

It cannot be used to discover theorems, but only to prove them.

If we have a propositional function $P(n)$, and we want to prove that $P(n)$ is true for any natural number $n$, we do the following:

1. Show that $P(0)$ is true. (basis step)
2. Show that if $P(n)$ then $P(n + 1)$ for any $n \in \mathbb{N}$. (inductive step)
3. Then $P(n)$ must be true for any $n \in \mathbb{N}$. (conclusion)

Example ("Gauss"):

$$1 + 2 + \ldots + n = n (n + 1)/2$$

1. Show that $P(0)$ is true. (basis step)

   For $n = 0$ we get $0 = 0$. True.

2. Show that if $P(n)$ then $P(n + 1)$ for any $n \in \mathbb{N}$. (inductive step)

   
   \[
   \begin{align*}
   1 + 2 + \ldots + n & = n (n + 1)/2 \\
   1 + 2 + \ldots + n + (n + 1) & = n (n + 1)/2 + (n + 1) \\
   & = (n + 1) (n/2 + 1) \\
   & = (n + 1) (n + 2)/2 \\
   & = (n + 1) ((n + 1) + 1)/2
   \end{align*}
   \]

3. Then $P(n)$ must be true for any $n \in \mathbb{N}$. (conclusion)

   $1 + 2 + \ldots + n = n (n + 1)/2$ is true for all $n \in \mathbb{N}$.

End of proof.

End of first lecture!