COS 210 - Theoretical Computer Science

Dr Nils Timm

Theory of Computation

Overall question:

- What are the capabilities and limitations of computers?

Related questions:

- What is a computational problem?
- Can a certain problem be solved by a computer?
  I.e. can we construct an algorithm that solves the problem?
- If the problem can be solved, how efficient can it be solved?

Examples

Problem 1: Given an arbitrary natural number n, is n prime?

- Can be solved in polynomial time

Problem 2: Given an encrypted document without the decryption key, decrypt the document.

- Can be solved, but may take billions of years for modern encryption schemes.

Problem 3: Given an arbitrary software program S and input I, does S terminate on imput I?.

- Can not be solved by a computer.

Focus of this Course

We want to mathematically study the problem-solving capabilities and limitations of computers.

For this we need:

- **Abstract models** of computers and algorithms
- **Formal languages** can be used to define computational problems
- **Proof techniques** that allow to reason about the complexity of problems

Topic Overview

- **Mathematical foundations** and theorem **proof techniques**
- **Computational problems**
  - Decision problems, search problems, optimisation problems
- **Automata theory**
  - Study of computational models
  - Finite Automata, Context-Free Grammars, Turing Machines
- **Complexity theory**
  - What makes some problems *"hard"* and others *"easy"* to solve?
- **Computability theory**
  - Is a computational problem *"solvable"* or *"unsolvable"*?

Learning Objectives

After completing this course you should have the knowledge and skills to

- understand the fundamental concepts of Theory of Computation (formal languages, automata, grammars, machines)
- define computational problems as formal languages
- construct computational models corresponding to languages
- perform transformations between the different types of models
- prove complexity and decidability properties of poblems

Staff, Course Platform and Textbook

Staff:

Name	Responsibility	Contact
Nils Timm (Course coordinator)	lectures	ntimm@cs.up.ac.za
Steven Jordaan (Assistant Lecturer)	tutorials	u18074848@tuks.co.za

Course platform:

- Course will be handled via **ClickUP** (announcements, study guide, release of material, tests, homework submission)

PDF textbook:

- **Introduction to Theory of Computation.** A Maheshwari and M Smid, 2017

Schedule

Session	Day	Time	Venue
Lecture 1	Monday	11:30 - 12:20	IT 2-26
Lecture 2	Wednesday	08:30 - 09:20	IT 2-26
Tutorial	Friday	15:30 - 16:20	IT 2-26

- Tutorial sessions will start in Week 2 of the semester.
- In weeks where a class test will be written there won't be a tutorial session because class tests are scheduled during the tutorial hours.

Mark Calculation

- Semester mark 60% + exam mark 40%
- Semester mark calculation:

Assessment	**Remark**	**Weight**
Worksheets	Best $N-1$ out of N	10%
Class Tests	Best 2 out of 3	30%
Semester Test 1		30%
Semester Test 2		30%

- Worksheets: weekly homework to be submitted via ClickUP
- Class tests: ClickUP tests written from home or an open lab
- Semester tests: ClickUP tests written in the Informatorium labs
- Exam:
  - 3-hour ClickUP test written in the Informatorium labs
  - exam entrance requires semester mark of at least 40%

Preliminary Test and Exam Dates

Activity	**Date**	**Time**
Semester Test 1	Thursday, 30 March	17:30
Semester Test 2	Thursday, 11 May	17:30
Sick Test	TBA	TBA
Final Exam	Monday, 20 June	07:30
Supplementary Exam	TBA	TBA

These tests and exams will be written in the Informatorium labs

Mathematical Foundations: Sets

The concept of a set is fundamental to any theoretical study. **A set is a collection of well-defined objects.** The most common sets you have encountered are:

- Boolean truth values $\mathbb{B} = \{0, 1\}$
- Natural numbers $\mathbb{N} = \{0, 1, 2, 3, \cdots\}$
- Integers $\mathbb{Z} = \{\cdots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \cdots\}$
- Rational numbers $\mathbb{Q} = \{\frac{m}{b} : m \in \mathbb{Z}, b \in \mathbb{Z}, b \neq 0\}$
- Real numbers $\mathbb{R} = \{\cdots, \sqrt{2}, \cdots, \pi, \cdots\}$

Mathematical Foundations: Sets

Operations on sets:

Let A and B be arbitrary sets then

- **Union**
$$A \cup B = \{x : x \in A \text{ or } x \in B\}$$
- **Intersection**
$$A \cap B = \{x : x \in A \text{ and } x \in B\}$$
- **Difference**
$$A \backslash B = \{x : x \in A \text{ and } x \notin B\}$$
- **Complement**
$$\bar{A} = \{x : x \notin A\}$$

Can we construct the **Difference** operator from the other operators?

Can we construct the **Difference** operator from the other operators?

- Observe that

$$A \backslash B = \{x : x \in A \text{ and } x \notin B\} = A \cap \bar{B}$$

Mathematical Foundations: Sets

Sometimes the result of an operation is a set of a different structure

- **Cartesian Product** of A and B

$$A \times B = \{(x, y) : x \in A \text{ and } y \in B\}$$

- For example let $A = \{1, 2, 3, 4\}$, and $B = \{0, 1\}$ then

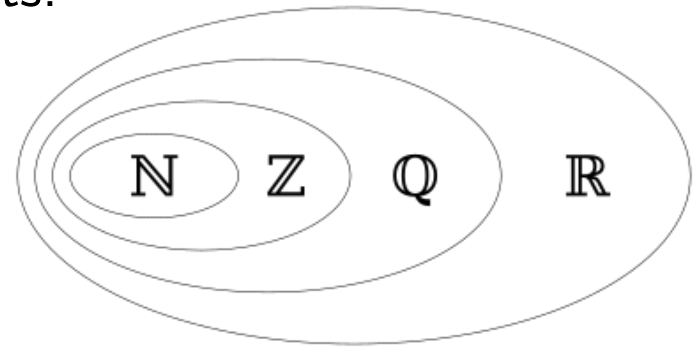
$$A \times B = \{(1, 0), (1, 1), (2, 0), (2, 1), (3, 0), (3, 1), (4, 0), (4, 1)\}$$

  Each element of the resulting set is a pair.

Subset, **Proper Subset** and **Equal** relationships between sets:

- **Subset**
  $A \subseteq B$ if for every $x \in A$ then $x \in B$
  - Every set A is a subset of itself, $A \subseteq A$
  - The empty set is a subset of every set, $\emptyset \subseteq A$
- **Proper Subset**
  $A \subset B$ if for every $x \in A$ then $x \in B$ but there is at least one $y \in B$ such that $y \notin A$
- **Equal**
  $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$

Subset, **Proper Subset** and **Equal** relationships between sets:



- **Subset**
  $A \subseteq B$ if for every $x \in A$ then $x \in B$
  - Every set A is a subset of itself, $A \subseteq A$
  - The empty set is a subset of every set, $\emptyset \subseteq A$
- **Proper Subset**
  $A \subset B$ if for every $x \in A$ then $x \in B$ but there is at least one $y \in B$ such that $y \notin A$
- **Equal**
  $A = B$ if and only if $A \subseteq B$ and $B \subseteq A$

Mathematical Foundations: Sets

The **power set** operator is defined based on the subsets

$$\mathcal{P}(B) = \{A : A \subseteq B\}$$

"the set of all subsets"

- $\emptyset \in \mathcal{P}(B)$
- $B \in \mathcal{P}(B)$
- For example if $B = \{1, 2, 3\}$ then

$$\mathcal{P}(B) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$$

where $\emptyset$ is the empty set (a set with no elements)

Mathematical Foundations: Binary Relations

- Formally a **binary relation** over two sets A and B is a subset R of $A \times B$.
- Binary relations can be used to select elements of $A \times B$ that satisfy a certain criterion
- For example

$$R = \{(x, y) : x < y, (x, y) \in \mathbb{N} \times \mathbb{N}\}$$

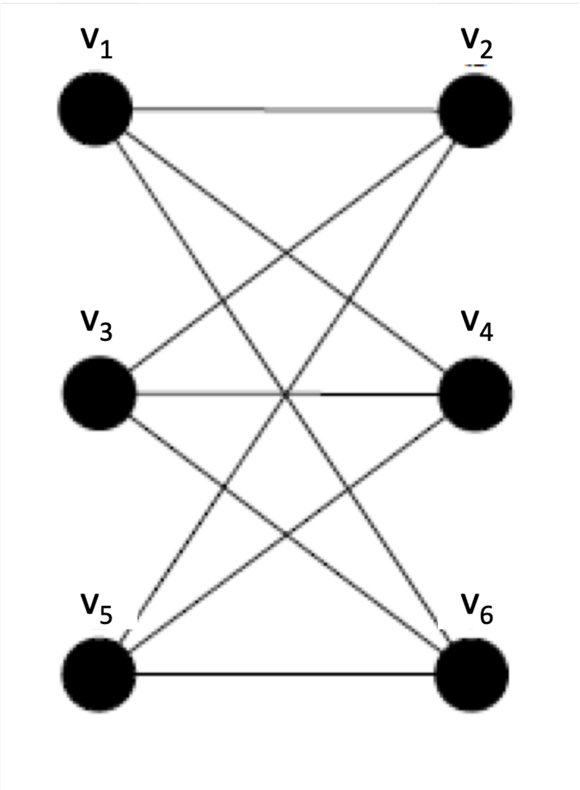
is a binary relation corresponding to all points in an $\mathbb{N} \times \mathbb{N}$ coordinate system above the line $y = x$.

Mathematical Foundations: Graphs

A **graph** is the pair $G = (V, E)$ where

- V is the set of vertices
- $E \subseteq \{(v, v') : v \neq v' \quad v, v' \in V\}$

Example:



$$V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

$$E = \{(v_1, v_2), (v_1, v_4), (v_1, v_6), (v_2, v_3), (v_2, v_5), (v_3, v_4), (v_3, v_6), (v_4, v_5), (v_5, v_6)\}$$

Mathematical Foundations: Boolean logic

NOT	AND	OR	XOR
$\neg 0 = 1$	$0 \wedge 0 = 0$	$0 \vee 0 = 0$	$0 \oplus 0 = 0$
$\neg 1 = 0$	$0 \wedge 1 = 0$	$0 \vee 1 = 1$	$0 \oplus 1 = 1$
	$1 \wedge 0 = 0$	$1 \vee 0 = 1$	$1 \oplus 0 = 1$
	$1 \wedge 1 = 1$	$1 \vee 1 = 1$	$1 \oplus 1 = 0$

equivalence	implication
$0 \leftrightarrow 0 = 1$	$0 \rightarrow 0 = 1$
$0 \leftrightarrow 1 = 0$	$0 \rightarrow 1 = 1$
$1 \leftrightarrow 0 = 0$	$1 \rightarrow 0 = 0$
$1 \leftrightarrow 1 = 1$	$1 \rightarrow 1 = 1$

Mathematical Foundations: Boolean logic

Boolean satisfiability problem:

Given a Boolean formula F, does there exist an assignment of Boolean values to the variables of F that makes the formula *true*?

Example:

$$x \wedge (y \vee \neg z) \wedge \neg y$$

This formula a satisfiable for the assignment $x \mapsto 1, y \mapsto 0, z \mapsto 0$

Many computational problems of practical relevance, for instance deadlock detection, can be reduced to the Boolean satisfiability problem