Theory of Computer Science

A1. Organizational Matters

Gabriele Röger

University of Basel

February 20, 2023

About this Course

Main Objectives

We would like to understand what can be computed

- in principle: decidability/computability
- efficiently: complexity theory

Uncomputable Problems?

Consider functions whose inputs are strings:

```
def program_returns_true_on_input(prog_code, input_str):
    ...
    # returns True if prog_code run on input_str returns True
    # returns False if not
```

Uncomputable Problems?

Consider functions whose inputs are strings:

```
def program_returns_true_on_input(prog_code, input_str):
    ...
    # returns True if prog_code run on input_str returns True
    # returns False if not

def weird_program(prog_code):
    if program_returns_true_on_input(prog_code, prog_code):
        return False
    else:
        return True
```

Uncomputable Problems?

Consider functions whose inputs are strings:

```
def program_returns_true_on_input(prog_code, input_str):
    ...
    # returns True if prog_code run on input_str returns True
    # returns False if not

def weird_program(prog_code):
    if program_returns_true_on_input(prog_code, prog_code):
        return False
    else:
        return True
```

What is the return value of `weird_program`
if we run it on its own source code?

Why should we Study the Theory of Computation?

- Theory is useful
  - If we want to solve a problem with a computer we need to know what is achievable. Computable? Tractable?
  - If the problem is not tractable, we might want to consider alternatives, e.g. a tractable variant or an approximation.
  - Some theoretical concepts have practical applications, e.g. regular expressions.

Why should we Study the Theory of Computation?

- Theory is useful
  - If we want to solve a problem with a computer we need to know what is achievable. Computable? Tractable?
  - If the problem is not tractable, we might want to consider alternatives, e.g. a tractable variant or an approximation.
  - Some theoretical concepts have practical applications, e.g. regular expressions.
- Theory is fun
  - Often like a brainteaser: E.g. how can we solve a problem exploiting a solver for some other problem?

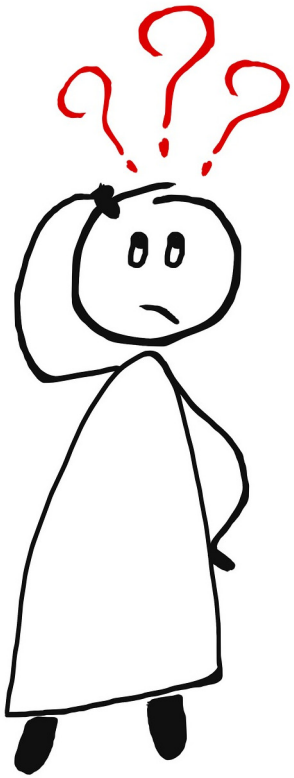
Content: Theoretical Foundations of Computer Science

A. background
   ▷ mathematical foundations and proof techniques
B. automata theory and formal languages
   (Automatentheorie und formale Sprachen)
   ▷ What is a computation?
C. Turing computability (Turing-Berechenbarkeit)
   ▷ What can be computed at all?
D. complexity theory (Komplexitätstheorie)
   ▷ What can be computed efficiently?
E. more computability theory (mehr Berechenbarkeitheorie)
   ▷ Other models of computability

Learning Goals

- understanding the capabilities and limitations of computers
- working with formal systems
  - comprehending formal definitions and theorems
  - precise formulation of definitions, theorems and proofs
  - analyzing formal problems precisely

Questions about the Course



Questions?

Organizational Matters

People

Lecturer

Gabi Röger

- email: gabriele.roeger@unibas.ch
- office: room 04.005, Spiegelgasse 1

Assistant

Salomé Eriksson

- email: salome.eriksson@unibas.ch
- office: room 04.003, Spiegelgasse 5

People

Tutors

Renato Farruggio

- email: renato.farruggio@stud.unibas.ch

Esther Mugdan

- email: esther.mugdan@unibas.ch

Mario Tachikawa

- email: mario.tachikawa@stud.unibas.ch

Time & Place

Lectures

- Monday: 14:15–16:00, Biozentrum, lecture hall U1.141
- Wednesday: 16:15–18:00, Alte Universität, lecture hall -101

Exercise Sessions (starting March 6/7)

- Monday 16:15–18:00
  - With Renato: ~~Kollegienhaus, lecture hall 116~~ Biozentrum, seminar room 02.054
  - With Esther: Alte Universität, lecture hall -201
- Tuesday 12:15–14:00
  - With Mario: Spiegelgasse 1, room U1.001

important: please send Salomé an email with your exercise partner and preferred slot until Wednesday 23:59 (February 22).

Exercises

Exercise sheets (homework assignments):

- mostly theoretical exercises
- on ADAM every Wednesday
- may be solved in groups of 2
- due Wednesday the following week
  (upload to Adam at `https://adam.unibas.ch/`)
- submission PDFs must be created with LaTeX
  → ADAM workspace: template and introduction to LaTeX

Exercises

Exercise sessions:

- discussion of previous exercise sheet (common problems)
- questions about current exercise sheet
- questions about the course
- if time: work on the homework assignment
- participation voluntary but highly recommended

Revised Course Format since 2022

- previously: 8 CP for lectures and exercises
- new: 6 CP main course + 2 CP for exercises
- separate enrolment and evaluation
- can and should be taken in parallel

Enrolment

- MOnA: `https://services.unibas.ch/`
- deadline: March 20
- better today for the course, so that you get all relevant emails and access to the ADAM workspace
- enrolment for exercise after we made the group assignment

Evaluation of Main Course (6 CP)

- written exam, 6 ECTS credits, graded 1-6
- 29 June 2023, 14:00-16:00
- admission to exam: no prerequisites
- must register for exam during April 3 – April 17
  $\rightsquigarrow$ see `https://philnat.unibas.ch/de/examen/`
- grade for course determined exclusively by the exam
- if you fail: one repeat attempt (within one year)

Evaluation of Main Course (6 CP)

- written exam, 6 ECTS credits, graded 1-6
- 29 June 2023, 14:00-16:00
- admission to exam: no prerequisites
- must register for exam during April 3 – April 17
  $\rightsquigarrow$ see `https://philnat.unibas.ch/de/examen/`
- grade for course determined exclusively by the exam
- if you fail: one repeat attempt (within one year)

Last lecture (May 31): Q&A session for exam preparation

Evaluation of Exercises (2 CP)

- pass/fail evaluation
- to pass the exercises, you need 50% of the exercise marks

Resources

- Adam: central starting point and exercises
  `https://adam.unibas.ch/`
- Website: course information, slides
- Discord: for your interaction with each other
  feel free to use a pseudonym

Course Material

course material:

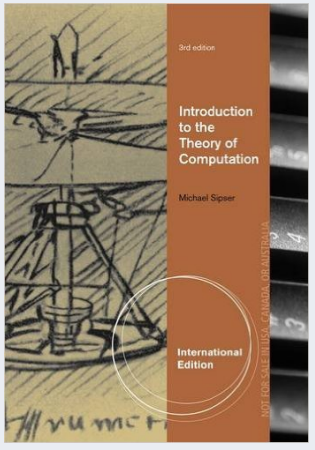
- slides (online)
- textbooks (see next slides)
- additional material on request

Course Material

Textbooks (English)

Introduction to the Theory of Computation
by Michael Sipser (3rd edition)

- covers most of the course
- also contains advanced topics
  beyond the scope of this course

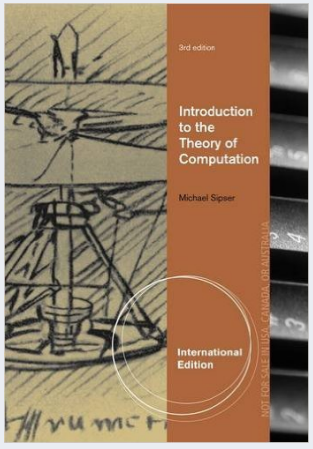


Course Material

Textbooks (English)

Introduction to the Theory of Computation
by Michael Sipser (3rd edition)

- covers most of the course
- also contains advanced topics beyond the scope of this course



Textbook (German)

Theoretische Informatik – kurz gefasst
by Uwe Schöning (5th edition)

- covers the course
- some concepts defined a bit differently (e.g. PDAs)

Prerequisites

- basic proof techniques
  (mathematical induction, proof by contradiction, ...)
- basic programming skills

Plagiarism

Plagiarism (Wikipedia)

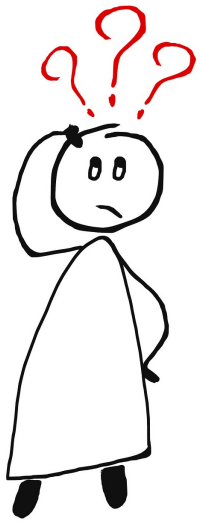
Plagiarism is the "wrongful appropriation" and "stealing and publication" of another author's "language, thoughts, ideas, or expressions" and the representation of them as one's own original work.

consequences:

- 0 marks for the exercise sheet (first time)
- exercises failed (second time)

if in doubt: check with us what is (and isn't) OK before submitting
exercises too difficult? we are happy to help!

Questions on Organization



Questions?