Theory of Computer Science

A1. Organizational Matters

Gabriele Röger

University of Basel

Organizational Matters

People

Lecturer

Gabi Röger

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

People

Tutors

Augusto B. Corrêa

- email: `augusto.blaascorrea@unibas.ch`
- office: room 04.001, Spiegelgasse 5

Florian Pommerening

- email: `florian.pommerening@unibas.ch`
- office: room 04.005, Spiegelgasse 1

Time & Place

Lectures

- Monday: 13:15–16:00
- Wednesday: 16:15–18:00
- live on Zoom

Time & Place

Exercise Sessions (starting March 8)

- group 1 (Augusto; in English)
- group 2 (Florian; in German)
- time: Monday 16:15–17:00
- on Zoom

important: please send Florian an email with your preferred language

until Wednesday 23:59 (March 3).

Revised Course Format since 2020

5 hours of lectures every week?!?

Revised Course Format since 2020

5 hours of lectures every week?!?

- more hands-on experience during the lectures
- bring pen & paper or tablet
- no increase of content
- overall time unchanged (now 5+1, previously 4+2)

Even More Revised Course Format in 2021

- Previously: Mathematical logic was part of the theory course
- Now: Covered in new course on Discrete Mathematics in CS
- We will focus on the standard curriculum and mostly use the freed time to gain a deeper understanding and more intution.

Online Course

- Adam: central starting point and exercises
- Website: course information, slides, additional material
- Zoom: lecture and exercise meetings
  please use your camera
- Discord: for your interaction with each other
  feel free to use a pseudonym
- Slido: feedback during lectures
  join at slido.com

Course Material

Textbook (German)

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

- covers most of the course

Uwe Schöning

Theoretische Informatik
– kurz gefasst

5. Auflage

HochschulTaschen*buch*

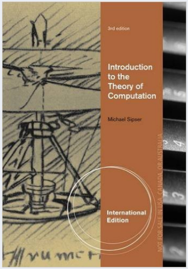
Spektrum

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



Target Audience

target audience:

- B.Sc. Computer Science, 4th semester
- B.A. Computer Science, 4th or 6th semester as an elective or if interested in M.Sc. Computer Science degree
- all other students welcome

prerequisites:

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

Enrolment

- MOnA: `https://services.unibas.ch/`
- deadline: March 29
- better today, so that you get all relevant emails

Exam

- written exam, 8 ECTS credits
- June 9, exact time and place TBA
- admission to exam: no prerequisites
- must register for exam during April 12 – April 26
  ⇝ see `https://philnat.unibas.ch/de/examen/`
- grade for course determined exclusively by the exam
- if you fail: one repeat attempt in FS 2022

Exercises

Exercise sheets (homework assignments):

- mostly theoretical exercises
- some programming exercises

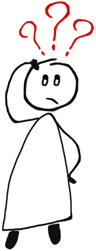
Exercise sessions:

- the tutors and you will decide together how to use the time. Some possibilities:
  - questions about exercise sheets
  - questions about the course
  - discussion of common problems
- participation voluntary but recommended

Exercises

- exercise sheets on ADAM every Wednesday
- may be solved in groups of arbitrary size (recommended: 2–3)
- due Wednesday the following week
  (upload to Adam at `https://adam.unibas.ch/`)
- scans must be legible (no photos, please)
- we appreciate LaTeX submissions

Questions on Organization



Questions?

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 odd_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 odd_program(prog_code):
    if program_returns_true_on_input(prog_code, prog_code):
        return False
    else:
        return True
```

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

Solution

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

Warning

"Wer's nicht gewohnt ist,
für den ist es ungewohnt."
(Prof. Dr. Th. Ottmann)
[If you are not used to it,
it may be unusual for you.]

Warning

“Wer’s nicht gewohnt ist,
für den ist es ungewohnt.”
(Prof. Dr. Th. Ottmann)
[If you are not used to it,
it may be unusual for you.]

What can you do?

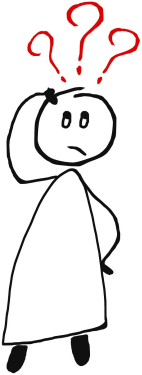
Warning

“Wer’s nicht gewohnt ist,
für den ist es ungewohnt.”
(Prof. Dr. Th. Ottmann)
[If you are not used to it,
it may be unusual for you.]

What can you do?

- stay on the ball
- do the exercises
- pay attention to details
- ask questions!

Questions about the Course



Questions?