



Theoretical Computer Science

Bridging Course

Introduction / General Info

–

Summer Term 2023

Fabian Kuhn

About the Course



Topics

- Foundations of theoretical computer science
- Introduction to logic

No lectures

- There are recordings which you are supposed to watch

Exercises

- There will be weekly exercises which you need to do
 - Doing the exercises is not mandatory, but highly recommended

Exam

- A oral exam at the end of the term
 - Details will be published on the course web page a.s.a.p.

About the course

What is the purpose of the course?

Who is it targeted to?

- The course is for incoming M.Sc. students who do not have the necessary theory background required by the M.Sc. program.
 - E.g., students who did not study computer science or students from more applied schools, ...

- All necessary information about the course will be published on http://ac.informatik.uni-freiburg.de/teaching/ss_23/tcs-bridging.php
 - Or go to my group's website: <http://ac.informatik.uni-freiburg.de>
 - Then follow teaching – summer term 2023 – TCS bridging course
- Please check the website for
 - Recordings and slides
 - Exercises and sample solutions
 - Pointers to additional literature (e.g., written lecture notes from an older version of this lecture)
 - Information about the exam
 - ...

There will be weekly exercise sheets:

- **Exercise sheets** are **published** at the latest **on Tuesday** on the website
- Exercises are **due after one week** on the **Tuesday at 12:15** before the exercise tutorial
 - If you want corrections / comments from your tutor
- Hand in your exercises on paper (in tutorial) or by email
- If you work in a group, the group needs to hand in one solution
 - Make sure that all students participate in solving & writing equally!
- After getting back your exercises, you can meet and discuss the exercises with your tutor
 - On Tuesdays or if additional help is necessary on request

Exercise Tutorials

Assistants for the course:

- Jonghyun Ham, jonghyun.ham@students.uni-freiburg.de
- Salwa Faour, salwa.faour@cs.uni-freiburg.de

Weekly Tutorials:

- There is a weekly tutorial on Tuesday from 12:15 – 14:00
 - The tutorials will be in-person (physical) in room 051-00-031
- In the tutorial, we discuss the upcoming exercise sheet and your solutions of the last exercise sheet
 - You are encouraged to actively participate in the tutorials and ask questions.
- Also ask your tutor if you have any questions!

The exercises are the most important part of the course!

- To pass the exam, it is important that you do the exercises
- If you feel comfortable with all the exercises, you should also be able to pass the exam

- When working in groups, make sure that you all participate in solving the questions and in writing the solutions!
 - You should all be able to explain your solutions to your tutor.

Course Topics

Foundations of Theoretical Computer Science

- Automata theory
- Formal languages, grammars
- Turing machines
- Decidability
- Computational complexity

Introduction to Logic

- Propositional logic
- First order logic

Purpose of the Course

Goal: Understand the **fundamental capabilities** and **limitations** of **computers**

- What does it mean to “compute”?
 - Automata theory
- What can be computed?
 - Theory on computability/decidability
- What can be computed efficiently?
 - Computational complexity

Meaning of “Computing”

Mathematical Models

- Turing machines 1930s
- Finite state automata 1940s
- Formal grammars 1950s

Practical Aspects

- Compute architectures 1970s
- Programming languages 1970s
- Compilers 1970s

Is My Function Computable?

Write an algorithm / computer program to compute it

- Can it compute the right answer for every instance?
- Does it always give an answer (in finite time)?
- Then you are done.

Otherwise, there are two options

- There is an algorithm, but you don't know it
- There is no algorithm \rightarrow the problem is unsolvable

Formally proving computability is sometimes hard!

- But you will learn how to approach this...

Is My Function Computable?

- Many “known” problems are solvable
 - Sorting, searching, knapsack, TSP, ...
- Some problems are not solvable
 - Halting problem
 - Gödel incompleteness theorem
- Don't try to solve unsolvable problems!

Can I Compute My Function Efficiently?



- Some problems are “easy”
 - Can we formally define what this means?
- **Complexity theory** is about this
 - Complexity classes, tools for checking membership
- It is important to know how hard a problem is!

- **Feasible problems:**
 - E.g., sorting, linear programming, LZW compression, primality testing, ...
 - Time to solve is polynomial in the size of the input
- **Problems that are considered infeasible**
 - Some scheduling problems, knapsack, TSP, graph coloring, ...
 - Important open question: “Is $P = NP$ ”?
- **Unfeasible problems**
 - Time exponential in input, e.g., quantified Boolean formula

Questions?
