Theoretical Computer Science (Bridging Course)

Introduction

Gian Diego Tipaldi

Time and Place

- Lecture
  - Today 08:00 – 10:00
- Exercises
  - Monday 10:00 – 12:00 (appointment)
- Building 52 – SR 02 017

People

- Dr. Gian Diego Tipaldi (lecturer)
  - Office: Building 79
  - Office hours: by arrangement (via email)
  - Email: tipaldi@cs.uni-freiburg.de
- Mr. Federico Boniardi (assistant)
  - Office: Building 79
  - Office hours: by arrangement (via email)
  - Email: boniardi@cs.uni-freiburg.de

Website

http://ais.informatik.uni-freiburg.de/

- Go to Teaching (Lehre) SS 2015
- Choose Theoretical computer science

- Syllabus
- Slides
- Exercise
- Additional material

Course Facts

- Course language
  - Lectures are given in English
  - Exercises are given in English
  - Exam will be in English

- Literature
  - Michael Sipser. *"Introduction to the theory of computation"*.
    PWS Publishing Co., Boston, MA, 1996

Course Content

- 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

What are the fundamental capabilities and limitations of computers?

Purpose of the Course

What are the fundamental capabilities and limitations of computers?

- What does it mean “to compute”?
- What can be computed?
- What can be computed efficiently?

Purpose of the Course

What are the fundamental capabilities and limitations of computers?

- What does it mean "to compute"?
  - Automata theory
- What can be computed?
  - Computability/Decidability theory
- What can be computed efficiently?
  - Computational complexity

The Meaning of "Compute"

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

- Practical aspects
  - Computer architectures 1970s
  - Programming languages 1970s
  - Compilers 1970s

Is my Function Computable?

- Write an algorithm to compute it
  - Can it compute every instance?
  - Will it always give you an answer?
  - Then you are done.
- If not, there are two choices
  - There is an algorithm but you don’t know
  - There exists no algorithm -> Unsolvable

- Formally prove computability is hard

Is my Function Computable?

- Many "known" problems are solvable
  - Sorting
  - Knapsack
- Other problems are not solvable
  - Halting problem
  - Gödel incompleteness theorem
- Don't try to solve unsolvable problems

Can I Compute it Efficiently?

- Some problems are “easy”
- Can we formally define it?

- Complexity theory comes to help
  - Complexity classes
  - Tools for checking class membership

- Important to know how hard it is

Can I Compute it Efficiently?

- Feasible problems
  - Sorting, linear programming, LZW
  - Time is polynomial in input
- Considered-unfeasible problems
  - Scheduling, Knapsack, TSP
  - Big open question: P=NP?
- Unfeasible problems
  - Quantified boolean formula
  - Time is exponential in input

Homework Assignment

- Available on Monday
  - On the website
- Due on Sunday one week after

- Solutions discussed on Monday
- Questions
  - Email to Federico or to me
  - Google Group

Homework Rules

- Group of max 2 people
  - both names, one submission

- Exam is on the same topics:
  Do the exercises!

Exam

- Written exam at the end
- Total points at the exam: 100
- Total points to pass: 50

Rule of thumb:

If you pass 50% of the exercises, you will pass the exam

Questions?