Theoretical Computer Science (TCS): Overview

Organization

Syllabus: should be in class materials

40% midterm + 40% exam + 10% homework + 10% presentation

Exam will only include material covered after midterm

2 Lectures + one Central Exercise + 1 TTF

Central Exercise = Lecture (maybe little slower)

4 major topics

- formal languages and their relation to automata
- computability
- logics and provability
- complexity theory

4 major topics

- formal languages and their relation to automata
- computability
- logics and provability
- complexity theory

Expressive power/comfort versus computational effort

- hierarchy of families of languages L
  with increasing expressive power

- deciding if an input w is a (well formed) word
  in L becomes more and more difficult

4 major topics

- formal languages and their relation to automata
- computability
- logics and provability
- complexity theory

Recursion Theory

- How to define computability ???
- why is our definition good?
- how to show, that something is not computable ?

4 major topics

- formal languages and their relation to automata
- computability
- logics and provability
- complexity theory

Logic

- what is a proof-system?
- what can be proven in such a system
- what is (provably) not provable?

4 major topics

- formal languages and their relation to automata
- computability
- logics and provability
- complexity theory

complexity theory

- models of computation and their relation

- what is computable with what resources?
  - time
  - space
- upper bounds
  - efficient algorithms
  - as you know a rich garden
- lower bounds on required resources
  - the wastelands
  - an oasis (and hope) here and there…

table of contents

table of contents

usual comment:

- impossible in 1 semester
- without prior knowledge

table of contents

3 main techniques

- simulation theorems between models of computation
- reducibility
- diagonalisation

usual comment:

- impossible in 1 semester
- without prior knowledge

table of contents

1. Regular expressions, finite automata and nondeterminism
2. Context free languages (including LR(k) languages) and pushdown automata
3. Context sensitive languages, type-0 languages and Turing machines
4. Recursive functions, computability and decidability
5. Undecidable problems and the recursion theorem
6. Proof Systems
7. Goedel's incompleteness theorems
8. Turing machine complexity classes
9. Complexity hierarchies
10. Speed up and Gap theorem
11. NP completeness
12. More about complete problems
13. Time versus space
14. Determinism versus nondeterminism
15. Applications of Kolmogorov complexity

usual comment:

- impossible in 1 semester
- without prior knowledge

3 main techniques

- simulation theorems between models of computation
- reducibility
- diagonalisation

as you well know:

simulation theorems are at the heart of system architecture correctness

hence some suggest:

one should start CS curriculum with TCS

table of contents

1. Regular expressions, finite automata and nondeterminism
2. Context free languages (including LR(k) languages) and pushdown automata
3. Context sensitive languages, type-0 languages and Turing machines
4. Recursive functions, computability and decidability
5. Undecidable problems and the recursion theorem
6. Proof Systems
7. Goedel's incompleteness theorems
8. Turing machine complexity classes
9. Complexity hierarchies
10. Speed up and Gap theorem
11. NP completeness
12. More about complete problems
13. Time versus space
14. Determinism versus nondeterminism
15. Applications of Kolmogorov complexity

usual comment:

- impossible in 1 semester
- without prior knowledge

2 main techniques

- simulation theorems between models of computation
- reducibility
- diagonalisation

as you well know:

simulation theorems are at the heart of system architecture correctness

hence some suggest:

one should start CS curriculum with TCS

you already know

models of computation

- finite automata
  - clocked circuits
  - MIPS
  - control automata
- abstract C machine
- CVM

you already know

models of computation

- finite automata
  - clocked circuits
  - MIPS
  - control automata
- abstract C machine
- CVM

nondeterminism

- MIPS + devices

you already know

models of computation

- finite automata
  - clocked circuits
  - MIPS
  - control automata
- abstract C machine
- CVM

nondeterminism

- MIPS + devices

simulation theorems

- processor correctness
- compiler correctness
- OS kernel correctness

you already know

models of computation

- finite automata
  - clocked circuits
  - MIPS
  - control automata
- abstract C machine
- CVM

nondeterminism

- MIPS + devices

simulation theorems

- processor correctness
- compiler correctness
- OS kernel correctness

formal language theory

- context free grammars
  - e.g. for C0
- Younger’s algorithm

you already know

models of computation

- finite automata
  - clocked circuits
  - MIPS
  - control automata
- abstract C machine
- CVM

nondeterminism

- MIPS + devices

simulation theorems

- processor correctness
- compiler correctness
- OS kernel correctness

formal language theory

- context free grammars
  - e.g. for C0
- Younger’s algorithm

recursion theory

- inductive definitions of +,*, exp,…
- primitive recursive functions
- Ackermann’s function
- a hint of diagonalisation

you already know

models of computation

- finite automata
  - clocked circuits
  - MIPS
  - control automata
- abstract C machine
- CVM

nondeterminism

- MIPS + devices

simulation theorems

- processor correctness
- compiler correctness
- OS kernel correctness

formal language theory

- context free grammars
  - e.g. for C0
- Younger’s algorithm

recursion theory

- inductive definitions of +,*, exp,…
- primitive recursive functions
- Ackermann’s function
- a hint of diagonalisation

the pebble game

- on trees

you already know

models of computation

- finite automata
  - clocked circuits
  - MIPS
  - control automata
- abstract C machine
- CVM

nondeterminism

- MIPS + devices

simulation theorems

- processor correctness
- compiler correctness
- OS kernel correctness

formal language theory

- context free grammars
  - e.g. for C0
- Younger’s algorithm

recursion theory

- inductive definitions of +,*, exp,…
- primitive recursive functions
- Ackermann’s function
- a hint of diagonalisation

the pebble game

- on trees

you have a flying start

at least you should have
if you remember

expect technical highlights

1. Regular expressions, finite automata and nondeterminism
2. Context free languages (including LR(k) languages) and pushdown automata
3. Context sensitive languages, type-0 languages and Turing machines
4. Recursive functions, computability and decidability
5. Undecidable problems and the recursion theorem
6. Proof Systems
7. Goedel’s incompleteness theorems
8. Turing machine complexity classes
9. Complexity hierarchies
10. Speed up and Gap theorem
11. NP completeness
12. More about complete problems
13. Time versus space
14. Determinism versus nondeterminism
15. Applications of Kolmogorov complexity

expect technical highlights

1. Regular expressions, finite automata and nondeterminism
2. Context free languages (including LR(k) languages) and pushdown automata
3. Context sensitive languages, type-0 languages and Turing machines
4. Recursive functions, computability and decidability
5. Undecidable problems and the recursion theorem
6. Proof Systems
7. Goedel's incompleteness theorems
8. Turing machine complexity classes
9. Complexity hierarchies
10. Speed up and Gap theorem
11. NP completeness
12. More about complete problems
13. Time versus space
14. Determinism versus nondeterminism
15. Applications of Kolmogorov complexity

here!