Exchange programme Vrije Universiteit

Vrije Universiteit Amsterdam - Exchange programme Vrije Universiteit - 2022-2023

Exchange

Vrije Universiteit Amsterdam offers many English-taught courses in a variety of subjects, ranging from arts & culture and social sciences, neurosciences and computer science, to economics and business administration.

The International Office is responsible for course approval and course registration for exchange students. For details about course registration, requirements, credits, semesters and so on, please visit the exchange programmes webpages.

Automata and Complexity

Course Code	X_401049
Credits	6.00
Period	P4
Course Level	300
Language Of Tuition	English
Faculty	Faculty of Science
Course Coordinator	dr. J. Endrullis
Examiner	dr. J. Endrullis
Teaching Staff	dr. J. Endrullis
Teaching method(s)	Lecture, Seminar

Course Objective

The first part of the course, on automata and languages, deals with the concepts of formal language, grammar, and automaton. Two types of languages are covered: regular and context-free languages. Regular languages in the form of regular expressions are ubiquitous in computer science. They are used in search queries and text manipulation. Context-free languages and grammars (e.g. the Backus–Naur form) are the prevailing standards for describing programming languages. We also discuss parsing algorithms to determine whether a given string is in a context-free language. The automata-theoretic counterparts for regular and context-free languages are finite automata and the more powerful pushdown automata.
The students will learn to:
(a) design finite automata and create regular expressions for a given regular language, [Knowledge and understanding] [Applying knowledge and understanding]
(b) apply algorithms to translate between finite automata, right-linear grammars, and regular expressions, [Knowledge and understanding] [Applying knowledge and understanding]
(c) apply algorithms to make automata deterministic and minimal, [Knowledge and understanding] [Applying knowledge and understanding]
(d) design pushdown automata and create context-free grammars for a given context-free language, [Knowledge and understanding] [Applying knowledge and understanding]
(e) apply algorithms to translate between pushdown automata and context-free grammars, [Knowledge and understanding] [Applying knowledge and understanding]
(f) use pumping lemmas to reason about whether a language is regular or context-free, [Applying knowledge and understanding] [Making judgements]
(g) apply algorithms for parsing context-free languages. [Knowledge and understanding] [Applying knowledge and understanding]

In the second part of the course, on computability theory, the central question is: what computations can be performed on a computer? To reason about computability, we introduce the mathematical model of Turing machines and discuss the Church-Turing thesis. We discuss examples of undecidable problems: the halting problem and the Post correspondence problem. It is shown how the undecidability of new problems can be shown by reduction from a known undecidable problem. Finally, we discuss a classification of decidable/computable problems into important complexity classes, notably P, NP, and NP-complete problems, together with the corresponding reduction arguments.
The students will learn to:
(h) reason whether a given problem is decidable (computable) or undecidable (not computable), [Making judgements]
(i) understand the classification of decidable problems in the complexity hierarchy (e.g. P, NP, EXP) of, and [Knowledge and understanding]

(j) reason about the complexity of a problem via the reduction to/from problems with known complexity. [Making judgements] [Communication] [Lifelong learning skills]

Course Content

This course treats automata & formal languages and computability theory. The student gets acquainted with important notions and algorithms regarding formal languages, automata, grammars, compilers, computability, and complexity.

This course addresses foundational questions in computer science:
- What can be computed? What are the limitations to what computers can do?
- How much time and memory does solving a problem require?
- What is a (programming) language?
- How can languages be recognized by computers (automata)?
- Which problems can be solved by what kinds of automata?

This course conveys the important idea that certain problems cannot be solved by computers. A computer scientist must be able to reason whether a given problem is computable (decidable) or not. Moreover, a computer scientist should be able to reason what language/complexity class a given problem belongs to, and hence what kind of automata/algorithms are needed to tackle the problem.

Additional Information Teaching Methods

4 hours per week lectures; 4 hours per week exercise classes.

Method of Assessment

The homework is mandatory for qualifying for the exam (70% of the homework points to qualify for the exam). In case at least 90% of the homework points is obtained, 0.5 bonus point is awarded for the final grade. At the end of the course there is a final exam.
The overall grade is the grade of the final exam plus the possibly 0.5 bonus point obtained for the homework. (The bonus is only added for students that pass the exam with a grade of at least 5.5.)
There is no resit opportunity for the homework.

Literature

Peter Linz, An Introduction to Formal Languages and Automata, Jones & Bartlett, 4th or 5th edition

Additional Information Target Audience

BSc Computer Science (year 3)