CS415-15 Decision Procedures

26/27

Department
Computer Science
Level
Undergraduate Level 4
Module leader
Dmitry Chistikov
Credit value
15
Module duration
10 weeks
Assessment
Multiple
Study location
University of Warwick main campus, Coventry

Description

Introductory description

In computer science, a special place in modelling belongs to mathematical logic. Given a yes/no question formulated in a logical language, a decision procedure produces a correct yes/no answer.

This module focuses on decision procedures for first-order logical theories. A common term for describing this field is Satisfiability Modulo Theories (SMT). Satisfiability (SAT) solvers and SMT solvers are software tools that implement decision procedures. They are used in practice to solve difficult combinatorial problems related to chip design, software and hardware verification, planning, and other applications.

Since the ability of these techniques to cope with problems arising in industry depends critically on decision procedures, this is a vibrant and prospering research subject for many researchers around the world, both in academia and in industry.

Intel and AMD, for example, are developing and using theorem provers and decision procedures as part of their efforts to build circuit verification tools with ever-growing capacity. Microsoft is developing and routinely using decision procedures in several code analysis tools. In 2022 Amazon reported running billions of SMT queries daily to offer their cloud service customers configurable and rigorously enforced security policies.

Module aims

The aim is for the students to become familiar with the modelling, logical, algorithmic and implementation aspects of decision procedures for first-order theories that are useful in the context of automated verification and reasoning, theorem proving, compiler optimization, synthesis, and so forth.

The theories covered in this module are all used in practice. We illustrate applications of each theory with examples representative of real problems, whether they may be verification of C programs, verification of hardware circuits, or optimizing compilers.

Outline syllabus

This is an indicative module outline only to give an indication of the sort of topics that may be covered. Actual sessions held may differ.

Decision procedures for propositional logic, equality logic and uninterpreted functions. SAT solvers, binary decision diagrams, small-domain instantiations.

Decision procedures for linear arithmetic, bit vectors, arrays and pointers. Simplex, branch and bound, bit-vector arithmetic, software verification.

Theories with quantifiers, combining theories, propositional encodings.

Implementation in C++, benchmarks, SMT solvers.

Learning outcomes

By the end of the module, students should be able to:

- Select and apply techniques from mathematical logic, linear algebra, algorithmic graph theory and other areas to construct decision procedures for a wide range of useful first-order theories.
- Recognise and compare theories with respect to their utility in modelling real decision problems, and their amenability to efficient solving in practice.
- Implement cutting-edge decision procedures on top of an extensible template, in an industry-standard language such as C++.
- Evaluate decision procedures using international repositories of relevant benchmarks.

Indicative reading list

Reading lists can be found in Talis

Subject specific skills

Interpretation of terms and formulas in logical theories.

Automation of reasoning with the use of decision procedures:

- Identification of an appropriate logic based on a problem description in natural language.

- Translation of problems from natural language into logical formulas (logical modelling).
- Application of decision procedures for theories and combinations of theories to logical formulas.

Analysis of decision procedures with respect to computational complexity.

Use and development of software for automated reasoning.

Transferable skills

Formalisation of problems. Identification of ambiguities and unknowns in problem description. Identification of constraints in problem description.

Algorithmic thinking. Use and development of software tools to solve problems.

Study

Study time

Type	Required
Lectures	20 sessions of 1 hour (13%)
Seminars	10 sessions of 1 hour (7%)
Practical classes	10 sessions of 1 hour (7%)
Private study	65 hours (43%)
Assessment	45 hours (30%)
Total	150 hours

Private study description

Concepts and algorithms covered in the module will be put into practice in module coursework (45 hours private study) as well as the timetabled lab sessions.

Students are also expected to engage in self-study to absorb content and prepare for the examination (65 hours).

Costs

No further costs have been identified for this module.

Assessment

You do not need to pass all assessment components to pass the module.

Students can register for this module without taking any assessment.

Assessment group D1

	Weighting	Study time	Eligible for self-certification
Practical assignment	30%	45 hours	No
A programming assignment which will target problem solving using decision procedures.			
CS415 Exam	70%		No
In-person examination			

- Answerbook Pink (12 page)

Assessment group R

	Weighting	Study time	Eligible for self-certification
CS415 Resit Exam	100%		No

- Answerbook Pink (12 page)

Feedback on assessment

- Individual written feedback.
- Group feedback in seminars.

Past exam papers for CS415

Availability

Courses

This module is Optional for:

- Year 5 of UCSA-G504 MEng Computer Science (with intercalated year)
- Year 4 of UCSA-G503 Undergraduate Computer Science MEng

This module is Option list A for:

- Year 4 of USTA-G304 Undergraduate Data Science (MSci)

This module is Option list B for:

- Year 4 of UCSA-G4G3 Undergraduate Discrete Mathematics
- Year 5 of UCSA-G4G4 Undergraduate Discrete Mathematics (with Intercalated Year)