Automated Reasoning in Generating Exam Sheets for Automated Deduction

Petra Hozzová, Laura Kovács, and Jakob Rath

TU Wien, Austria

Abstract. Amid the COVID-19 pandemic, distance teaching became the default in world-wide higher education, urging teachers and researchers to revise course materials into a more accessible online content for a diverse audience. Probably one of the hardest challenges in this new form of education came with online assessments of course performance, especially organizing and grading online written exams. In this short paper we focus on our teaching experience during our master's level course "Automated Deduction" at TU Wien, and report on the automated reasoning we developed for generating individual online exam sheets for students enrolled in the course. The algorithmic and rigorous logical reasoning developed within our course calls for exam sheets focused on problem solving and deductive proofs; as such exam sheets using test grids are not a viable solution for written exams within our course. We believe that the toolchain of automated reasoning tools we have developed for holding online written exams could be beneficial not only for other distance learning platforms, but also to researchers in automated reasoning by providing our community with a large set of randomly generated benchmarks in SAT/SMT solving and first-order theorem proving.

1 Motivation

online exams due to pandemic. we want to avoid collaboration between students during exam (or make it at least a bit harder), so each student gets their own exam sheet. etc...

2 todo

Challenge: problem instances should be different but of similar difficulty to make sure the exam is fair to students.

3 Method 1: Varying Templates and Fixed Patterns

3.1 SMT problem

relatively easy: provided template, do small random perturbations that don't change the solution

3.2 Ground superposition problem

todo

4 Method 2: Full Random Generation

more sophisticated: full random generation, filter out "too hard"/"too easy" instances. Note that we don't need very efficient implementation of filters since the instances are very small. so we can use naive satisfiability tests or model counting.

4.1 SAT problem

describe filters, and why they were chosen (aim for a challenging problem, but still solvable by hand).

problems: when restricting too much, the resulting formulas may end up boring. e.g. SAT formula with exactly one model, or no model

4.2 Superposition+Redundancy problem

todo