Is Your Combinatorial Search Algorithm Telling the Truth?

Ciaran McCreesh

ciaran.mccreesh@glasgow.ac.uk

University of Glasgow

How do you know whether your combinatorial search algorithm is implemented correctly? You could try testing it, but even if you're convinced you've done a thorough job, will anyone else believe you? Another possibility is "correct by construction" software created using formal methods—but these methods are far from being able to approach the complexity or performance of modern satisfiability or constraint programming solvers. In this talk I'll tell you about a third option, called *proof logging* or *certifying*. The idea is that, alongside a solution, an algorithm must produce a mathematical proof in a standard format that demonstrates that the solution is correct. This proof can be verified by an independent proof checker, which should be much simpler, and thus easier to trust. The key challenge in getting this to work is to find a proof language which is both simple to verify, and expressive enough to cover a wide range of solving techniques with very low overheads. It's not obvious that such a language should even exist, but I'll argue that cutting planes with a dominance-based extension rule might be exactly what we need: even though cutting planes has no notion of what vertices, graphs, or even integers are, it is strong enough to verify the reasoning used in state of the art algorithms for problems like subgraph isomorphism, clique, and maximum common connected subgraph, and even in constraint programming solvers.