Kleene Algebra — Project suggestions

Tobias Kappé

For each of these projects, I can suggest some papers to read. Your task will be to deliver a 30-minute lecture, including questions, on your chosen topic.

Let me just stress that none of these projects are set in stone. If you are interested in some specific topic not on this list, or if you want to focus more on one part of a suggested project, we can discuss that.

Kleene Algebra with Tests. We talked about KAT in the first half of the course. It turns out that decidability and completeness can also be proved for this extended version of KA. How do these proofs work, and how can we reuse the results that we saw in the last lecture to make them easier to understand?

Adding concurrency. One recent extension, proposed by Hoare and collaborators, is to add an operator for concurrent composition, yielding Concurrent Kleene Algebra (CKA). How does the model of this extension account for the added concurrency? And how can one recover decidability and completeness?

Guarded programs. We saw that KAT can encode traditional flow control constructs. But what if we study a language with **if** and **while** constructs as primitives? This fragment of KAT, called GKAT, has a number of fascinating properties. What makes GKAT different from KAT in terms of its expressivity?

Application: NetKAT. NetKAT is a programming language that can be used to program computer networks. As an extension of KAT, it admits a completeness theorem, as well as decidable equivalence. What are the consequences of these strong theoretical results for the application domain?

Advanced equivalence checking. The bisimulation checking algorithm that we discussed leaves some room for optimization. The family of *bisimulation up to* techniques allows one to optimize bisimulation checking for the automata we saw (and some generalizations). How fast can we make these algorithms?

Complexity. We know that equivalence of programs in Kleene Algebra is decidable. In fact, it turns out that this problem is quite general: it is PSPACE-complete. However, when we add some additional hypotheses to our axioms, the theory becomes undecidable. Where is the boundary between these two?

Hypotheses Can we add laws to Kleene Algebra that are specific to a particular set of primitive programs? *Kleene Algebra with Hypotheses* is a framework for adding additional laws to Kleene Algebra, while retaining decidability and completeness. How powerful is this framework, and where are its limits?