Project Proposal

Ben Little

Oct 23, 2023

Formalize Basic Complexity Results in Agda

I will look into existing complexity libraries for Coq, Agda, and other proof assistants. I will compare different models like *boolean*-valued predicates vs *type*-valued predicates. I will attempt to demonstrate some proofs we did in the problem sets, like showing that $\mathbf{C} \subseteq \mathrm{co}\mathbf{C} \implies \mathbf{C} = \mathrm{co}\mathbf{C}$ and the Time Hierarchy Theorem.

Some starting points for this research are

- Gäher, Lennard, and Fabian Kunze. "Mechanising complexity theory: the cook-Levin theorem in Coq." 12th International Conference on Interactive Theorem Proving (ITP 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
- Xu, Jian, Xingyuan Zhang, and Christian Urban. "Mechanising turing machines and computability theory in Isabelle/HOL." Interactive Theorem Proving: 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings 4. Springer Berlin Heidelberg, 2013.
- Catt, Elliot, and Michael Norrish. "On the formalisation of Kolmogorov complexity." Proceedings of the 10th ACM SIGPLAN International Conference on Certified Programs and Proofs. 2021.
- Forster, Yannick, Fabian Kunze, and Nils Lauermann. "Synthetic Kolmogorov Complexity in Coq." 13th International Conference on Interactive Theorem Proving (ITP 2022). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.
- Bauer, Andrej. "First steps in synthetic computability theory." Electronic Notes in Theoretical Computer Science 155 (2006): 5-31.
- Forster, Yannick, Dominik Kirst, and Gert Smolka. "On synthetic undecidability in Coq, with an application to the Entscheidungsproblem." Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs. 2019.

Gäher and Kunze prove the Cook-Levin theorem (all NP problems are p-time reducible to SAT) using a series of verified steps to compile decision problems written in L into SAT instances. They define the generic NP problem for L GenNP that asks, "given an L program s, a size k for the witness, and number of steps t, does there exist an input of size no greater than k for which s halts in no more than t steps?" They show that GenNP is NP-hard, i.e. for every L program in NP there exists a poly-time reduction to GenNP. They then prove there exists a poly-time reduction to the generic problem for Turing Machines TMGenNP, and then eventually that TMGenNP poly-time reduces to SAT.