Summary:

Superposition for Higher-Order Logic

Automated theorem provers are computer programs that automatically find mathematical proofs. For many years, their development has been divided into two branches: provers for first-order logic and provers for higher-order logic.

First-order logic is the simpler language to express mathematics and allowed the first-order logic community to develop highly efficient provers. The superposition calculus emerged already in the early 1990s and is arguably still the most efficient procedure for first-order logic today, especially when reasoning with equations.

Higher-order logic is a more expressive language, enabling us to directly express concepts such as big mathematical operators (e.g., Σ, ∏, ∫). The higher-order branch of automated theorem proving developed a variety of strategies to keep their logic’s high expressivity under control. Unfortunately, many concepts that led to success in first-order logic (such as term orders and redundancy criteria) seemed not to fit into the theory of higher-order reasoning.

The work presented in my thesis brings these two worlds together by generalizing the superposition calculus to higher-order logic. The guiding principle has been that the procedure should operate on higher-order logic without giving up what has been successful for first-order logic since the 1990s. In particular, the procedure is complete—i.e., the search space is explored entirely, without missing a possible proof. The procedure avoids exploring the same area multiple times (using a redundancy criterion) or paths that will not lead to a proof (using a term order). Together with colleagues, I have implemented the procedure. Based on an early version of this implementation, our prover won the higher-order division of the 2020 edition of the CASC prover competition.