HOList: an Open-Source, AI-Oriented Environment for Tactic-Based Theorem Proving

Kshitij Bansal, Christian Szegedy

Given the fundamental nature of mathematics and its foundational aspects for almost all scientific disciplines, the capability for high level formal mathematical reasoning is both an important practical task as well as one of the most challenging case studies in AI. Higher-order logic is a prime contender for (automatic) formalizing of any kind of mathematical content.

In this talk, we give the general outline of a new open-source environment named HOList that is designed for machine learning and AI researchers to interact with tactic-based higher-order proof assistants. The goal of our system is to encapsulate and abstract away the technical details of dealing with the communication and proof search within proof assistants that were designed for human use. Our philosophy was to create an open-source, easy to install and use, modular environment that allows AI agents to interact with such logic proof assistants. The simple, stable APIs and the modular nature of HOList allows for researchers to share code at various levels of abstraction. Our current implementation can only interface with the HOL Light proof assistant, but we rely on a very thin, stateless, language agnostic RPC-based ProofAssistantService as a communication medium between the proof-search system and the assistant, which could be implemented with low overhead for other proof assistants as well. That would allow the rest of HOList to be trained to act as a theorem proving agent for other similar proof assistants as well. A more significant effort went into instrumenting HOL Light with tactic execution tracing mechanisms allowing for imitation learning in order to bootstrap reinforcement learning systems.

Our system comes with a simple theorem prover, named DeepHOL, that utilizes deep neural networks for predicting the most efficient tactic invocations. This includes selecting the correct tactic parameters from a large corpus of theorem parameters and works as a premise selection mechanism. Again, we have a thin API between the proof search graph and the AI model. In essence the search graph can act as an environment of AI agents. HOList's architecture is aimed to make it simple to integrate with other machine learning models.

We also publish a large corpus of human proof logs derived from the core and complex libraries of HOL Light. Furthermore we transformed this data into TF-Examples which is a file format for storing data which can be easily and efficiently read by TensorFlow models. In order to allow for transparent benchmarking, this data was split into well defined training, validation, and testing sets to evaluate the performance of various machine learning models trained with imitation

learning. The training splits are done at the proof level, so performance on each set can also be evaluated by number theorems proved using the prover with the learned models. We have also imported the whole formal proof of the Kepler conjecture for further benchmarking purposes.

We share all the training data and theorem databases which allows for benchmarking various AI algorithms in the final theorem proving context. Our goal is to allow AI researchers to focus on the machine learning algorithmic aspects of theorem proving without needing to understand the technical details of the underlying logic or proof assistant. To further lower the complexity of setup, we provide docker images for both the proof assistant and the proof-search part of the system that interact with each other via RPC calls. We have also taken performance considerations into account by addressing the slow startup time of HOL Light by creating mechanisms for “cheating in” theorems and definitions quickly. However, this methodology runs into the risk of programming errors leading to inconsistent theories as not all theorem objects are fully checked by the trusted kernel. To prevent this, we created proof verifiers that can fully check the correctness of proof logs by running through the whole theory together with their new proofs through the trusted kernel as a final verification step.

By creating a simple, efficient system for higher order theorem proving, we lower the barriers of entry for AI researchers to study this important area of application. In addition, by providing stable well defined APIs and benchmarks, we hope to foster collaboration between research teams and to allow for more transparent and comparable evaluation of various approaches in this domain.