Implementing an evaluator for mini-ML in ATS: a case of programming with theorem proving*

Hongwei Xi

Boston University

Abstract *Applied Type System* ($\mathcal{ATS}$) is a recently developed framework for designing and formalizing type systems in support of practical programming. The development of $\mathcal{ATS}$ is an attempt to overcome some acute problems with Pure Type System ($\mathcal{PTS}$) that make it difficult, especially, in the presence of dependent types to accommodate many common realistic programming features such as general recursion, recursive types, effects (e.g., exceptions, references, input/output), etc. The key salient feature of $\mathcal{ATS}$ lies in a complete separation of the statics, in which types are formed and reasoned about, from the dynamics, in which programs are constructed and evaluated. With this separation, it is no longer possible for a program to occur in a type as is otherwise allowed in $\mathcal{PTS}$. Currently, a programming language ATS with a highly expressive type system rooted in the framework $\mathcal{ATS}$ is under active development. In ATS, there is a (limited) theorem proving component ATL/LF that allows the programmer to encode (inductive) proofs as (total recursive) functions, supporting a programming paradigm that combines programs with proofs. In this paper, we present in ATS the implementation of an evaluator for mini-ML, demonstrating an interesting case of programming with theorem proving. A particularly significant point we make is that the type system of the ATS actually guarantees the correctness of this implementation. We are also to stress some important differences between programming with theorem proving and extracting programs from proofs (constructed via theorem proving).

* Partly supported by NSF grant no. CCR-0229480