Some practical considerations about the use of the Curry-Howard Isomorphism on a industrial size case study

Simão Melo de Sousa
DI-UBI, Covilhã, Portugal
desousa@di.ubi.pt

We report on the use of Curry-Howard based proof assistants in the context of formal verification of large scale systems.

The present case study is the formal verification of correctness properties of the JavaCard virtual machine (the Java platform for smart cards). In this context we focus our attention on typing properties checked by the ByteCode Verifier (BCV). Bytecode verification is one of the key security functions of the JavaCard architecture. Its correctness is often cast relatively to a virtual machine, called *defensive*, that performs checks at run-time, and an *offensive* one that does not, and can be summarized as stating that the two machines coincide on programs that pass bytecode verification.

In this context, proof systems that implement the Curry-Howard isomorphism enjoy, in our opinion, some important properties like direct handling of proof objects or extraction mechanisms. This is exactly the kind of feature that leads us to choose the Coq proof assistant. However their use in formal verification of large systems raise some practical, but important, considerations. We can summarize them by (1) Are such proof assistants scaling up? (2) Are the benefits of the Curry-Howard isomorphism sufficient?

Our conclusions, shared by many other teams involved in large scale formal verification, are that such proof systems lack of tool support for large proof. For instance, when proving a particular property of the analyzed system, the proof effort often needs to focus on only a particular aspect of the system. Unfortunately, without modifications of the model, the proof has to take into account the system as a whole. A possible solution is to infer from the original specification a simpler model, in the sense of the *abstract interpretation*. Despite the fact that such transformations form the base of large scale formal verification, no proof system provides tool support for them.

Facts like this one lead us to define a toolset called JaKarTa. Its objective is to provide an interactive environment for transforming virtual machines and for verifying the correctness of these transformations in proof assistants.