A Formalised Theory of Turing Machines in Isabelle/HOL

Xu Jian, Xingyuan Zhang
PLA University of Science and Technology Nanjing, China

Christian Urban
King's College London, UK

***Abstract*—Isabelle/HOL is an interactive theorem prover based on classical logic. While classical reasoning allow users to take convenient shortcuts in some proofs, it precludes *direct* reasoning about decidability: every boolean predicate is either true or false because of the law of excluded middle. The only way to reason about decidability in a classical theorem prover, like Isabelle/HOL, is to formalise a concrete model for computation. In this paper we formalise Turing machines and relate them to register machines.**

***Keywords*-Turing Machines, Decidability, Isabelle/HOL;**

I. Introduction

We formalised in earlier work the correctness proofs for two algorithms in Isabelle/HOL—one about type-checking in LF [4] and another about deciding requests in access control [5]. The formalisations uncovered a gap in the informal correctness proof of the former and made us realise that important details were left out in the informal model for the latter. However, in both cases we were unable to formalise in Isabelle/HOL computability arguments about the algorithms. The reason is that both algorithms are formulated in terms of inductive predicates. Suppose P stands for one such predicate. Decidability of P usually amounts to showing whether $P \lor \neg P$ holds. But this does *not* work in Isabelle/HOL, since it is a theorem prover based on classical logic where the law of excluded middle ensures that $P \lor \neg P$ is always provable no matter whether P is constructed by computable means. The same problem would arise if we had formulated the algorithms as recursive functions, because internally in Isabelle/HOL, like in all HOL-based theorem provers, functions are represented as inductively defined predicates.

The only satisfying way out is to formalise a theory of computability. Norrish provided such a formalisation for the HOL4 theorem prover. He choose the λ-calculus as the starting point for his formalisation, because of its "simplicity" [3, Page 297]. Part of his formalisation is a clever infrastructure for reducing λ-terms. He also established the computational equivalence between the lambda-calculus and recursive functions. Nevertheless he concluded that it would be appealing to have formalisations of more operational models of computations such as Turing machines or register machines. One reason is that many proofs in the literature refer to them. He noted however that in the context of theorem provers [3, Page 310]:

> *"If register machines are unappealing because of their general fiddliness, Turing machines are an even more daunting prospect."*

In this paper
[1]
Our formalisation follows [2]

Contributions:

II. Wang Tiles

Used in texture mapings - graphics

III. Related Work

The most closely related work is by Norrish. He bases his approach on lambda-terms. For this he introduced a clever rewriting technology based on combinators and de-Bruijn indices for rewriting modulo β-equivalence (to keep it manageable)

References

[1] A. Asperti and W. Ricciotti. Formalizing Turing Machines. In *Proc. of the 19th International Workshop on Logic, Language, Information and Computation (WoLLIC)*, volume 7456 of *LNCS*, pages 1–25, 2012.

[2] G. Boolos and R. C. Jeffrey. *Computability and Logic (2. ed.)*. Cambridge University Press, 1987.

[3] M. Norrish. Mechanised Computability Theory. In *Proc. of the 2nd Conference on Interactive Theorem Proving (ITP)*, volume 6898 of *LNCS*, pages 297–311, 2011.

[4] C. Urban, J. Cheney, and S. Berghofer. Mechanizing the Metatheory of LF. *ACM Transactions on Computational Logic*, 12:15:1–15:42, 2011.

[5] C. Wu, X. Zhang, and C. Urban. ??? Submitted, 2012.