Relating Computational Effects by $\top\top$-Lifting

Shin-ya Katsumata

Research Institute for Mathematical Sciences (RIMS), Kyoto University,
Kitashirakawa Oiwakecho, Sakyoku, Kyoto 606-8502, Japan
sinya@kurims.kyoto-u.ac.jp

When we have two implementations of a programming language, we are naturally interested in knowing a relationship between these implementations. Suppose that we obtain two relationships on 1) data representations in two implementations and 2) behaviours of side-effects in two implementations. Then the question is whether the obtained relationships are respected by every program, that is, for every program P, when related data are supplied as input, the execution of P in two implementations raises related side-effects.

We consider this question in the following theoretical setting:

- For programming languages, we employ λ_c calculi extended with algebraic operations. We view them as idealised call-by-value functional programming languages.
- For implementations, we employ Moggi's monadic semantics of λ_c-calculi.

We present a sufficient condition for a given set of relationships for values and computations to be respected by every program. This condition is natural, and applicable to any λ_c-calculus with algebraic operations, monadic semantics and relationships under consideration.

The proof of the condition being sufficient hinges on the technique called categorical $\top\top$-lifting. It is a semantic formulation of Lindley and Stark's leapfrog method [3, 4], and constructs logical relations for monads. This construction takes a parameter, and by varying it we can derive various logical relations. In the proof of the sufficiency, we supply the relationship on computations as the parameter—this is the key to achieve the generality of the condition.

If time permits, I will talk about other applications of the categorical $\top\top$-lifting. This talk is based on [1, 2].

References

[1] S. Katsumata. A semantic formulation of $\top\top$-lifting and logical predicates for computational metalanguage. In L. Ong, ed., *Proc. CSL 2005*, *Lect. Notes Comput. Sci.*, v. 3634, pp. 87–102. Springer, 2005.

[2] S. Katsumata. Relating computational effects by $\top\top$-lifting. *Inf. Comput.*, v. 222, pp. 228–246, 2013.

[3] S. Lindley. Normalisation by Evaluation in the Compilation of Typed Functional Programming Languages. PhD thesis, University of Edinburgh, 2004.

[4] S. Lindley, I. Stark. Reducibility and $\top\top$-lifting for computation types. In P. Urzyczyn, ed., *Proc. TLCA 2005*, *Lect. Notes Comput. Sci.*, v. 3461, pp. 262–277. Springer, 2005.