Game semantics of the Safe λ-calculus

Abstract for the PRG Conference 2006

William Blum

September 3, 2006

The *safety condition* has been introduced in [4] as a syntactic restriction for higher-order recursion schemes (grammars) that constrains the order of the variables occurring in the grammar equations. The authors were able to prove that the Monadic Second Order (MSO) theory of the term tree generated by a safe recursion scheme of any order is decidable[1].

When transposed to the λ-calculus, the safety condition gives rise to the *Safe λ-calculus*, a strict sub-language of the λ-calculus. A first version appeared in the technical report [2]. We propose a more general and simpler version where term types are not required to be homogeneous ([3]). A noteworthy feature of the Safe λ-calculus is that no variable capture can occur when performing substitution and therefore it is unnecessary to rename variables when computing β-reductions.

Little is known about the Safe λ-calculus and there are many problems that have yet to be studied concerning its computational power, the complexity classes that it characterises, its interpretation under the Curry-Howard isomorphism and its repercussion on game semantics ([1]). We have contributed to give an answer to the latter by proving the following result:

Theorem 1 *The pointers in the game semantics of safe simply-typed terms can be recovered uniquely from the underlying sequences of moves.*

Giving a game-semantic account of Safety is complicated by the fact that it is a syntactic restriction whereas Game Semantics is by essence a syntax-independent semantics. The solution consists in making an explicit correspondence between some syntactical representation of a term and its game denotation. Our approach is based on ideas recently introduced in [5]: a term is represented by the abstract syntax tree of its η-long normal form, referred as the *computation tree*. A computation is represented by a justified sequence of nodes of the computation tree respecting some formation

[1]In fact it has been shown recently in [5] that it is also true for unsafe recursion schemes.

rules and called *traversal*. A traversal permits to achieve a *local computation* of β-reductions as opposed to global approaches that compute β-redexes by performing substitutions. A notable property is that the *P-view* (in the game-semantic sense) of a traversal is a path in the computation tree.

We prove the *Correspondence Theorem* stating that traversals are just representations of the uncovering of plays of the game denotation of the term. The nodes of the traversals that are "internal" to the computation are eliminated by an appropriate *reduction* operation. This leads to an isomorphism between the strategy denotation of a term and the set of reductions of traversals over its computation tree.

To prove Theorem 1 we first introduce the notion of *incrementally-justified strategies* and show that pointers are uniquely reconstructible for such strategies. We then introduce the notion of *incrementally-bound computation trees* and prove, using the Correspondence Theorem, that incremental-binding coincides with incremental-justification. Finally, we show that safe simply-typed terms in β-normal form have incrementally-bound computation trees.

We define Safe IA to be the Safe λ-calculus augmented with the constants of Idealized Algol (IA) as well as a family of combinators Y_A for every type A. We show that terms of the Safe PCF fragment are denoted by incrementally-justified strategies and we give the key elements for a possible extension to full Safe IA.

References

[1] S. Abramsky and G. McCusker. Game semantics. In *Logic and Computation: Proceedings of the 1997 Marktoberdorf Summer School*. Springer-Verlag, 1998. Lecture notes.

[2] K. Aehlig, J.G. de Miranda, and C.-H. L. Ong. Safety is not a restriction at level 2 for string languages. Technical report, University of Oxford, 2004.

[3] W. Blum. Transfer thesis. University of Oxford, August 2006.

[4] T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pushdown trees are easy. In *FOSSACS*, pages 205–222. Springer, 2002. LNCS Vol. 2303.

[5] C.-H. L. Ong. On model-checking trees generating by higher-order recursion schemes. In *Proceedings of LICS*. Computer Society Press, 2006.