Game semantics of the Safe lambda-calculus

William Blum

Oxford University, Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD, UK.
William.Blum@comlab.ox.ac.uk

1 The safety condition

The *safety condition* has been introduced in [5] as a syntactic restriction for higher-order recursion schemes (grammars) that constrains the order of the variables occurring in the grammar equations. The authors of [5] 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 [3], 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 simpler and more general version where term types are not required to be homogeneous [4]. 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 game-semantic characterisation [1]. Our contribution concerns the last problem.

2 The correspondence theorem

The difficulty in giving a game-semantic account of Safety lies in the fact that it is a syntactic restriction whereas Game Semantics is by essence a syntax-independent semantics. The solution consists in finding a particular syntactical representation of terms on which the plays of the game denotation can be represented. To achieve this, we use ideas recently introduced in [6]: a term is canonically represented by the abstract syntax tree of its η-long normal form, referred as the *computation tree*. A computation is described by a justified sequence of nodes of the computation tree respecting some formation rules and called a *traversal*. Traversals permit us to model β-reductions without altering the structure of the computation tree via substitution. We prove the following result:

Theorem 1 (Correspondence theorem) *The set of traversals of the computation tree is isomorphic to the set of uncovered plays of the game denotation of the term.*

In other words, traversals are just representations of the uncovering of plays of the strategy denoting the term. By defining an appropriate *reduction* operation which eliminates traversal nodes that are "internal" to the computation, we obtain an isomorphism between the strategy denotation of a term and the set of reductions of traversals of its computation tree.

[1] In fact it has been shown in [6] that it is also true for unsafe recursion schemes.

3 Game-semantic characterisation

We introduce the notions of *incrementally-justified strategies* and *incrementally-bound computation trees*. Using the Correspondence Theorem, we show that for β-normal terms the computation tree of a term is incrementally-bound if and only if its strategy denotation is incrementally-justified. We have the following theorem:

Theorem 2 (Game-semantic characterisation of safety) *Safe simply-typed terms in β-normal form have incrementally-bound computation trees. Reciprocally, a closed term in η-long normal form with an incrementally-bound computation trees is safe.*

Since pointers in the plays of *incrementally-justified strategies* are by definition uniquely reconstructible, we obtain the following corollary:

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

4 Extension to Safe Idealized Algol

We define Safe IA to be the Safe λ-calculus augmented with the constants of Idealized Algol (IA) [8] as well as a family of combinators Y_A for every type A. We show that terms of the Safe PCF [7] fragment are denoted by incrementally-justified strategies and we give the key elements for a possible extension to full Safe IA.

References

1. Samson Abramsky and Guy McCusker. Game semantics. In *Logic and Computation: Proceedings of the 1997 Marktoberdorf Summer School*. Springer-Verlag, 1998. Lecture notes.
2. Klaus Aehlig, Jolie G. de Miranda, and C.-H. Luke Ong. Safety is not a restriction at level 2 for string languages. Technical report, University of Oxford, 2004.
3. Hendrik Pieter Barendregt. *The Lambda Calculus – Its Syntax and Semantics*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland, 1984.
4. William Blum. Transfer thesis. University of Oxford, August 2006.
5. T. Knapik, D. Niwiński, and P. Urzyczyn. Higher-order pushdown trees are easy. In *FOSSACS*, pages 205–222. Springer, 2002. LNCS Vol. 2303.
6. C.-H. Luke Ong. On model-checking trees generating by higher-order recursion schemes. In *Proceedings of LICS*. Computer Society Press, 2006.
7. Gordon D. Plotkin. LCF considered as a programming language. *Theor. Comput. Sci.*, 5(3):225–255, 1977.
8. John C. Reynolds. The essence of algol. In J. W. de Bakker and J. C. van Vliet, editors, *Algorithmic Languages*, pages 345–372. IFIP, North-Holland, Amsterdam, 1981.