Nominal Game Semantics

Andrzej S. Murawski
University of Warwick

Nikos Tzevelekos
Queen Mary University of London

the essence of knowledge
Boston — Delft

Foundations and Trends® in Programming Languages

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
United States
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is

A. S. Murawski and N. Tzevelekos. *Nominal Game Semantics.* Foundations and Trends® in Programming Languages, vol. 2, no. 4, pp. 191–269, 2015.

This Foundations and Trends® issue was typeset in LaTeX using a class file designed by Neal Parikh. Printed on acid-free paper.

ISBN: 978-1-68083-107-8

Foundations and Trends® in Programming Languages
Volume 2, Issue 4, 2015

Editorial Board

Editorial Scope

Topics

Foundations and Trends® in Programming Languages publishes survey and tutorial articles in the following topics:

- Abstract interpretation
- Compilation and interpretation techniques
- Domain specific languages
- Formal semantics, including lambda calculi, process calculi, and process algebra
- Language paradigms
- Mechanical proof checking
- Memory management
- Partial evaluation
- Program logic
- Programming language implementation
- Programming language security
- Programming languages for concurrency
- Programming languages for parallelism
- Program synthesis
- Program transformations and optimizations
- Program verification
- Runtime techniques for programming languages
- Software model checking
- Static and dynamic program analysis
- Type theory and type systems

Information for Librarians

Foundations and Trends® in Programming Languages, 2015, Volume 2, 4 issues. ISSN paper version 2325-1107. ISSN online version 2325-1131. Also available as a combined paper and online subscription.

Foundations and Trends® in Programming Languages
Vol. 2, No. 4 (2015) 191–269

DOI: 10.1561/2500000017

Nominal Game Semantics

Andrzej S. Murawski
University of Warwick

Nikos Tzevelekos
Queen Mary University of London

Contents

Abstract

These tutorial notes present *nominal game semantics*, a denotational technique for modelling higher-order programs.

A. S. Murawski and N. Tzevelekos. *Nominal Game Semantics*. Foundations and Trends® in Programming Languages, vol. 2, no. 4, pp. 191–269, 2015.
DOI: 10.1561/2500000017.

1

Introduction

Game semantics is a branch of denotational semantics that uses the metaphor of game playing to model computation. The game models of PCF [5, 21, 35] constructed in the 1990s have led to an unprecedented series of *full abstraction* results for a range of functional/imperative programming languages. A result of this kind characterises contextual equivalence between terms semantically, i.e. equality of denotations coincides with the fact that terms can be used interchangeably in any context. As such, full abstraction results can be said to capture the computational essence of programs.

The fully abstract game models from the 1990s covered a plethora of computational effects, contributing to a general picture referred to as *Abramsky's cube* [8]: by selectively weakening the combinatorial conditions on plays of the games, one was able to increase the expressivity of the games and capture desired computational effects.

Although those works successfully constructed models of state [7, 6, 4, 9], the techniques used to interpret reference types did not make them fully compatible with what constitutes the norm in languages such as ML or Java. In particular, references were modelled through a form of indirection originating in the work of Reynolds [39], namely by assuming that $\mathsf{ref}\,\theta = (\theta \to \mathsf{unit}) \times (\mathsf{unit} \to \theta)$. The approach led to identification of references with pairs of arbitrary reading $(\mathsf{unit} \to \theta)$ and writing $(\theta \to \mathsf{unit})$ functions. While this view is elegant and certainly comprises the range of behaviours corresponding to references, it does not enforce a relationship between reading and writing, as witnessed by the presence of the product type. This causes a significant strengthening of the semantic universe used for modelling references and, consequently, many desirable equivalences are not satisfied in the model. For example, the interpretation of $(x := 0; x := 1)$ is different from that of $x := 1$ and, similarly, for $x := !x$ and $()$. We list the interpretations below using the terminology of [6].

$x := 0; x := 1$	run write(0) ok write(1) ok done
$x := 1$	run write(1) ok done
$x := !x$	run read i write(i) ok done
$()$	run done

Thus, for the first term, the semantic translation treats both updates as observable events and therefore both are recorded in the game play.[1] This immediately distinguishes semantically the first term from the second one, for which only a single update is recorded. On the other hand, the translation of the third term is more verbose, registering calls to both the read and write methods of x, even though the computational content of the term is in fact that of the skip command $()$ in the modelled language.

To prove full abstraction in this setting, it is then necessary to enrich the syntax with terms that will populate the whole semantic space of references. Such terms are often referred to as *bad variables*, because they are objects of reference type equipped with potentially unrelated reading and writing methods. These terms, if used by the context, can distinguish the pairs of terms discussed above. For instance, a context that instantiates x to a bad variable with divergent reading and writing capabilities will be able to distinguish $x := !x$ from $()$. Nonetheless, that solution is not entirely satisfactory as the bad-variable construct breaks standard expectations for references. Moreover, one would hope to be

[1] In effect, write(0) and write(1) represent calls to the write method of reference x, while ok's correspond to returns of that method.

able to carve the model in such a way that it matches the modelled language, instead of extending the language to match the model.

The bad-variable problem can be seen as the result of modelling a generative effect (the creation and use of references) by equating it with the product of its observable handling methods.[2] Nominal game semantics is a recent branch of game semantics that makes it possible to model generative effects in a more direct manner, by incorporating *names* (drawn from an infinite set) as atomic objects in its constructions. In particular, it can model reference types without bad variables by using names to interpret references. The names are embedded in moves and also feature in stores that are carried by moves in the game. Intuitively, the stores correspond to the observable part of program memory. For example, the two pairs of terms discussed above can be modelled by the following two nominal plays respectively.

$$a^{\{(a,i)\}}\ \star^{\{(a,1)\}} \qquad\qquad a^{\{(a,i)\}}\ \star^{\{(a,i)\}}$$

Here a stands for an arbitrary name, i.e. the collection of plays is stable with respect to name permutations. Formally, the objects studied in nominal game semantics (moves, plays, strategies) live in nominal sets [12].

Since 2004, the nominal approach has led to a series of new full abstraction results. The languages covered are the ν-calculus [3] (purely functional language with names), $\lambda\nu$ [25] (a higher-order language with storage of untyped names), Reduced ML [31] (a higher-order language with integer-valued storage), RefML [32] (higher-order references) and Middleweight Java [34]. Nominal game semantics has also been used to model Concurrent ML [26] and exceptions [34].

Structure of the tutorial

Our tutorial is meant to complement existing introductory literature to game semantics [1, 8, 19, 16], which highlighted the then new structural components necessary to model higher-order computation, e.g. arenas,

[2]Similar issues arise when modelling exceptions in this way, i.e. as products of raise/handle functions [24].

justification pointers, innocence. In contrast, we shall particularly focus on explaining the nominal content of our games. We hope the material has been written in a way that will make it accessible to readers familiar with standard denotational semantics and types, e.g. [10, 17, 40].

We begin our exposition with Chapter 2 covering the basics of nominal sets. In Chapter 3 we introduce the programming language of study, called GroundML. GroundML is a higher-order language with references capable of storing integers, reference to integers, references to references to integers and so on. In Chapter 5 we shall present the game model of GroundML in full detail. Before that, in Chapter 4, we focus on a fragment of GroundML that, for the sake of simplicity, features only integer-valued references and restricted higher-order types. Because ToyML is simpler, we can give a more direct and elementary presentation of its game semantics, which we hope will help the reader to make a transition to the full-blown model of the following section.

References

[1] S. Abramsky. Semantics of interaction. In A. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, pages 1–32. Cambridge University Press, Cambridge, 1997.

[2] S. Abramsky, D. R. Ghica, A. S. Murawski, and C.-H. L. Ong. Algorithmic game semantics and component-based verification. In *Proceedings of SAVBCS: Specification and Verification of Component-Based Systems, Workshop at ESEC/FASE*, Technical Report 03-11, pages 66–74. Department of Computer Science, Iowa State University, 2003.

[3] S. Abramsky, D. R. Ghica, A. S. Murawski, C.-H. L. Ong, and I. D. B. Stark. Nominal games and full abstraction for the nu-calculus. In *Proceedings of LICS*, pages 150–159. IEEE Computer Society Press, 2004.

[4] S. Abramsky, K. Honda, and G. McCusker. Fully abstract game semantics for general references. In *Proceedings of IEEE Symposium on Logic in Computer Science*, pages 334–344. Computer Society Press, 1998.

[5] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.

[6] S. Abramsky and G. McCusker. Call-by-value games. In *Proceedings of CSL*, volume 1414 of *Lecture Notes in Computer Science*, pages 1–17. Springer-Verlag, 1997.

[7] S. Abramsky and G. McCusker. Linearity, sharing and state: a fully abstract game semantics for Idealized Algol with active expressions. In P. W. O'Hearn and R. D. Tennent, editors, *Algol-like languages*, pages 297–329. Birkhaüser, 1997.

[8] S. Abramsky and G. McCusker. Game semantics. In H. Schwichtenberg and U. Berger, editors, *Logic and Computation*. Springer-Verlag, 1998. Proceedings of the 1997 Marktoberdorf Summer School.

[9] S. Abramsky and G. McCusker. Full abstraction for Idealized Algol with passive expressions. *Theoretical Computer Science*, 227:3–42, 1999.

[10] R. L. Crole. *Categories for Types*. Cambridge University Press, 1993.

[11] M. Gabbay and D. R. Ghica. Game semantics in the nominal model. *Electr. Notes Theor. Comput. Sci.*, 286:173–189, 2012.

[12] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002.

[13] Murdoch James Gabbay. *A Theory of Inductive Definitions with Alpha-Equivalence*. PhD thesis, University of Cambridge, 2001.

[14] D. R. Ghica and G. McCusker. Reasoning about Idealized Algol using regular expressions. In *Proceedings of ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 103–115. Springer-Verlag, 2000.

[15] D. R. Ghica and N. Tzevelekos. A system-level game semantics. *Electr. Notes Theor. Comput. Sci.*, 286:191–211, 2012.

[16] Dan R. Ghica. Applications of game semantics: From program analysis to hardware synthesis. In *Proceedings of LICS*, pages 17–26. IEEE Computer Society, 2009.

[17] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. MIT Press, 1992.

[18] K. Honda and N. Yoshida. Game-theoretic analysis of call-by-value computation. *Theoretical Computer Science*, 221(1–2):393–456, 1999.

[19] J. M. E. Hyland. Game semantics. In A. Pitts and P. Dybjer, editors, *Semantics and Logics of Computation*, pages 131–182. Cambridge Univ. Press, 1997.

[20] J. M. E. Hyland and C.-H. L. Ong. Pi-calculus, dialogue games and PCF. In *Proc. 7th* ACM *Conf. Functional Prog. Lang. Comp. Architecture*, pages 96 – 107. ACM Press, 1995.

[21] J. M. E. Hyland and C.-H. L. Ong. On Full Abstraction for PCF: I. Models, observables and the full abstraction problem, II. Dialogue games and innocent strategies, III. A fully abstract and universal game model. *Information and Computation*, 163(2):285–408, 2000.

[22] Guilhem Jaber. Operational nominal game semantics. In *Proceedings of FOSSACS*, pages 264–278. Springer, 2015.

[23] A. Jeffrey and J. Rathke. Java Jr: Fully abstract trace semantics for a core Java language. In *Proceedings of ESOP*, volume 3444 of *Lecture Notes in Computer Science*, pages 423–438. Springer, 2003.

[24] J. Laird. A fully abstract games semantics of local exceptions. In *Proceedings of 16th IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 2001.

[25] J. Laird. A game semantics of local names and good variables. In *Proceedings of FOSSACS*, volume 2987 of *Lecture Notes in Computer Science*, pages 289–303. Springer-Verlag, 2004.

[26] J. Laird. Game semantics for higher-order concurrency. In *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 417–428, 2006.

[27] J. Laird. A fully abstract trace semantics for general references. In *Proceedings of ICALP*, volume 4596 of *Lecture Notes in Computer Science*, pages 667–679. Springer, 2007.

[28] J. Laird. A game semantics of names and pointers. *Annals of Pure and Applied Logic*, 151:151–169, 2008.

[29] E. Moggi. Computational lambda-calculus and monads. In *Proceedings of IEEE Symposium on Logic in Computer Science*, pages 14–23. Computer Society Press, 1989.

[30] A. S. Murawski, S. J. Ramsay, and N. Tzevelekos. Game semantic analysis of equivalence in IMJ. submitted, 2015.

[31] A. S. Murawski and N. Tzevelekos. Full abstraction for Reduced ML. In *Proceedings of FOSSACS*, volume 5504 of *Lecture Notes in Computer Science*, pages 32–47. Springer-Verlag, 2009.

[32] A. S. Murawski and N. Tzevelekos. Algorithmic nominal game semantics. In *Proceedings of ESOP*, volume 6602 of *Lecture Notes in Computer Science*, pages 419–438. Springer-Verlag, 2011.

[33] A. S. Murawski and N. Tzevelekos. Algorithmic games for full ground references. In *Proceedings of ICALP*, volume 7392 of *Lecture Notes in Computer Science*, pages 312–324. Springer, 2012.

[34] A. S. Murawski and N. Tzevelekos. Game semantics for interface middleweight java. In *POPL*, pages 517–528, 2014.

[35] H. Nickau. *Hereditarily Sequential Functionals: A Game-Theoretic Approach to Sequentiality*. PhD thesis, Universität-Gesamthochschule-Siegen, 1996.

[36] Andrew M. Pitts. *Nominal Sets: Names and Symmetry in Computer Science*. Cambridge University Press, New York, NY, USA, 2013.

[37] J. Power and E. Robinson. Premonoidal categories and notions of computation. *Mathematical Structures in Computer Science*, 7:453–468, 10 1997.

[38] John Power and Hayo Thielecke. Closed Freyd- and kappa-categories. In *Proceedings of ICALP*, pages 625–634. Springer, 1999.

[39] J. C. Reynolds. The essence of Algol. In J. W. de Bakker and J.C. van Vliet, editors, *Algorithmic Languages*, pages 345–372. North Holland, 1981.

[40] J. C. Reynolds. *Theories of Programming Languages*. Cambridge University Press, 1998.

[41] Ulrich Schöpp. *Names and Binding in Type Theory*. PhD thesis, University of Edinburgh, 2006.

[42] N. Tzevelekos. Full abstraction for nominal general references. *Logical Methods in Computer Science*, 5(3), 2009.