Nominal Game Semantics

Nikos Tzevelekos

Queen Mary University of London

Game semantics has been developed since the 1990s as a denotational paradigm capturing observational equivalence in functional languages with imperative features. While initially introduced for PCF variants, the theory can nowadays express effectful languages ranging from ML fragments and Java programs to C-like code. In this talk we present recent advances in devising game models for effectful computation. Central in this approach is the use of names for representing in an abstract fashion different forms of notions and effects, such as references, higher-order values and polymorphism. We moreover look at automata models relevant to nominal games and how can they be used for model checking program equivalence. The material of this talk is presented in more detail in the joint tutorial [26].

1 Games for programming languages

Game semantics is a branch of denotational semantics that uses the metaphor of game playing to model computation. The game models of PCF [4, 12, 28] 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* [7]: 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 [6, 5, 3, 8], 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 [29], 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 $()$.

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. Nonetheless, that solution is not entirely satisfactory as the bad-variable construct breaks standard expectations for references. Moreover, one would hope to be able to carve the model in such a way that it matches the modelled language, instead of extending the language to match the model.

2 Nominal games

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. Similar issues arise when modelling exceptions in this way, i.e. as products of raise/handle functions [17]. 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. Names can be created fresh, compared for equality and passed around between program terms. They 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 [9].

Nominal game semantics has allowed for more satisfactory models of languages with computational effects. Since 2004, the nominal approach has led to a series of new full abstraction results. Some of the languages covered are the ν-calculus [2] (purely functional language with names), $\lambda\nu$ [18] (a higher-order language with storage of untyped names), Reduced ML [22] (a higher-order language with integer-valued storage), RefML [23] (higher-order references) and Middleweight Java [24]. Nominal games have also been used to model Concurrent ML [19] and exceptions [25].

Algorithmic game semantics

Game semantics provides a handle for attacking equivalence problems. One might wonder to what extent it can be used to automate the process. This direction was initiated and first pursued in classical game models, for fragments of Idealized Algol, where it was observed that by restricting type disciplines to low orders it was possible to concretely represent term strategies by means of finite-state or pushdown automata [10, 1]. We have investigated this problem for fragments of ML with ground references [23, 27] and Interface Middleweight Java [21]. Our technique is based on automata over infinite alphabets. This class is a natural fit for representing nominal plays due to the presence of names, drawn from an infinite alphabet.

Operational game semantics

The operational flavour of nominal games, combined with its concrete representation of names, leads one to consider connections to trace models for higher-order programs with names as examined e.g. in [16, 20, 11]. It turns out that games can be given a fully operational presentation and can be seen essentially as open trace models with composition as a primitive operation. This has been done in the specific case of games for higher-order references [13], but we can now envisage a general applicability of the approach. We have in particular developed operational game models for higher-order languages with polymorphism [14, 15]. There, names are given an additional semantic role, namely that of modelling abstract types and polymorphic values, thus imposing polymorphic parametricity on the model.

References

1. 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.
2. 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.
3. 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.
4. S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF. *Information and Computation*, 163:409–470, 2000.
5. 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.
6. 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.
7. 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.
8. S. Abramsky and G. McCusker. Full abstraction for Idealized Algol with passive expressions. *Theoretical Computer Science*, 227:3–42, 1999.
9. M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2002.
10. 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.
11. D. R. Ghica and N. Tzevelekos. A system-level game semantics. *Electr. Notes Theor. Comput. Sci.*, 286:191–211, 2012.
12. 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.
13. Guilhem Jaber. Operational nominal game semantics. In *Proceedings of FOSSACS*, pages 264–278. Springer, 2015.

14. Guilhem Jaber and Nikos Tzevelekos. Trace semantics for polymorphic references. In *Proceedings of LICS*, pages 585–594. ACM, 2016.
15. Guilhem Jaber and Nikos Tzevelekos. A trace semantics for system F parametric polymorphism. In *Proceedings of FOSSACS*, volume 10803 of *Lecture Notes in Computer Science*, pages 20–38. Springer, 2018.
16. 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.
17. 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.
18. 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.
19. J. Laird. Game semantics for higher-order concurrency. In *FSTTCS*, volume 4337 of *Lecture Notes in Computer Science*, pages 417–428, 2006.
20. 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.
21. A. S. Murawski, S. J. Ramsay, and N. Tzevelekos. Game semantic analysis of equivalence in IMJ. submitted, 2015.
22. 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.
23. 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.
24. A. S. Murawski and N. Tzevelekos. Game semantics for interface middleweight java. In *POPL*, pages 517–528, 2014.
25. Andrzej S. Murawski and Nikos Tzevelekos. Game semantics for nominal exceptions. In *Proceedings of FOSSACS*, volume 8412 of *Lecture Notes in Computer Science*, pages 164–179. Springer, 2014.
26. Andrzej S. Murawski and Nikos Tzevelekos. Nominal game semantics. *Foundations and Trends in Programming Languages*, 2(4):191–269, 2016.
27. Andrzej S. Murawski and Nikos Tzevelekos. Algorithmic games for full ground references. *Formal Methods in System Design*, 52(3):277–314, 2018.
28. H. Nickau. *Hereditarily Sequential Functionals: A Game-Theoretic Approach to Sequentiality*. PhD thesis, Universität-Gesamthochschule-Siegen, 1996.
29. 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.