Algorithmic Game Semantics and its Applications: Final Report

S. Abramsky C.-H. L. Ong

31 January 2006

1 Background and context

Game Semantics was introduced in the early nineties in the construction of the first fully complete model of Classical Multiplicative Linear Logic, and in the construction of the first syntax-independent fully abstract model of PCF. By the time of the project proposal – some ten years on, Game Semantics had emerged as a powerful paradigm for giving accurate semantics to a variety of programming languages and logical systems. It had been used to construct fully abstract models for a wide spectrum of programming languages covering features such as recursive types and polymorphism, non-local control operators, block-allocated local variables, general references, probabilistic and non-deterministic constructs, etc.

The aim of the project was to develop Game Semantics in a new, *algorithmic* direction, with a view to applications in computer-assisted verification and program analysis. Game Semantics has several features which make it very promising from this point of view. It provides a very *concrete* way of building *fully abstract* models. It has a clear operational content, while admitting *compositional methods* in the style of denotational semantics. The basic objects studied in Game Semantics are games, and strategies on games. Strategies can be seen as certain kinds of highly-constrained processes, hence they admit the same kind of automata-theoretic representations central to model checking and allied methods in computer-assisted verification. Moreover games and strategies naturally form themselves into rich mathematical structures which yield very accurate models of advanced high-level programming languages, as the various full abstraction results show. Thus the promise of this approach is to carry over the methods of model checking, which has been so effective in the analysis of circuit designs and communications protocols, to much more *structured* programming situations, in which data-types as well as control flow are important.

In relation to the extensive current activity in model checking and computer assisted verification, our approach is distinctive, being founded on a highly-structured *compositional* semantic model. This means that we can directly apply our methods to *program phrases* (i.e. terms-in-context with free variables) in a high-level language with procedures, local variables and data types; moreover, the soundness of our methods is guaranteed by the properties of the semantic models on which they are based. By contrast, most current model checking applies to relatively "flat" unstructured situations, in which the system being analyzed is presented as a transition system or automaton. Our aim is to build on the tools and methods which have been developed in the verification community, while exploring the advantages offered by our semantics-based approach.

2 Key advances

The starting point of Algorithmic Game Semantics was Ghica and McCusker's discovery (in ICALP'00 [9]) that the game semantics of the 2nd-order fragment of finitary Idealized Algol augmented by a while-construct (written $\mathsf{IA}_2{+}\mathsf{while}$) is regular. To what extent can Ghica and McCusker's order-2 result be extended? A challenging programme we set ourselves was: first, to determine the largest fragment of Idealized Algol (IA) for which observational equivalence remains decidable, and then to classify the fragment according to the complexity of the decision problem.

Complete classification of decidable IA

At low orders (up to 2), the pointers of a play (which are sequences of moves with pointers) are uniquely reconstructible from the underlying sequence of moves. From order 3 onwards, the pointer structure becomes an indispensable part of any representation of plays, and hence of strategies. In a LICS'02 paper [22], Ong has shown that by *modelling state explicitly* in games, observational equivalence of 3rd-order IA can be reduced to the problem of DPDA Equivalence, which is decidable (Sénizergues 1999). The explicit-state approach does not work at orders above 3. Indeed observational equivalence in finitary IA is undecidable in general. In a LICS'03 paper [14], Murawski has shown that the (game semantics of) 2nd, 3rd and 4th order fragments of IA define exactly regular, context-free and recursively enumerable languages. It follows that observational equivalence of IA_4 is undecidable.

Definition by recursion is a central programming construct. A natural way to measure off recursion into varying strengths is by the order of the result type of the fixpoint operator $\mathbf{Y}^A : (A \to A) \to A$; thus we write $\mathsf{IA}_i{+}\mathbf{Y}_j$ for IA_i augmented by $\mathbf{Y}^A$ such that $\mathsf{order}(A) \leq j$. In an ICALP'05 paper [19], Murawski, Ong and Walukiewicz have shown that $\mathbf{Y}_0$'s effect is essentially neutral: observational equivalence in $\mathsf{IA}_i{+}\mathbf{Y}_0$ is decidable iff it is in IA_i for all $i \geq 0$ (though the two have different complexities, as we shall see later). However $\mathbf{Y}_1$ breaks decidability [22].

We have obtained a **complete classification** of the fragment of IA for which observational equivalence is decidable, according to the complexity of the decision problem[1], as follows:

	pure	+**while**	$+\mathbf{Y}_0$	$+\mathbf{Y}_1$
IA_1	CONP	PSPACE	DPDA	(= $\mathsf{IA}_1{+}\mathbf{Y}_0$)
IA_2	PSPACE	PSPACE	DPDA	UND
IA_3	EXPTIME	EXPTIME	DPDA	UND
IA_4	UND	UND	UND	UND

In the Table above, UND means undecidable, and DPDA means "equivalent to DPDA Equivalence (in the sense of mutual reducibility)". The entry PSPACE (say) means that deciding observational equivalence of any two IA_2-terms (or any two (IA_2+while)-terms) *in normal form* is PSPACE-complete; similarly for the entries CONP and EXPTIME. The PSPACE-complete results [18] were obtained by Murawski using a PSPACE-construction of the deterministic finite automata representing strategy-denotations of (IA_2+while)-terms. The EXPTIME-complete results (in FOSSACS'05 [15]) were obtained by a representation of the game semantics in Alur and Madhusudan's *visibly pushdown automata*. The final piece of the jigsaw (in ICALP'05 [19]) that completes the classification was the "DPDA-complete" result for the fragments $\mathsf{IA}_i{+}\mathbf{Y}_0$ for $i = 1, 2$ and 3.

The call-by-value case of **Reduced ML** has been investigated by Murawski [17]. The main results are undecidability of observational equivalence at order 2, and a characterization of programs can be modelled using regular language in terms of type order and a certain notion of currying level.

[1] Namely, "for any two L-terms in β-normal form, are they observationally equivalent", where L is a given fragment of IA.

Generic polymorphism

Polymorphism is one of the most challenging programming language features to model. Abramsky and Jagadeesan [4] have developed a game semantics for polymorphism which captures the idea of *generic polymorphism* which had previously been proposed by Longo; the idea that *the same program is being used at differfent type instances*. Longo distilled a novel equational principle characterizing genericity, which was not satisfied by any known model of polymorphism. The model developed in [4] solved the problem, open for some 10 years, of giving a model (and in fact, a naturally semantically motivated one) which satisfied the Genericity axiom. The construction also had a number of other interesting features; in particular, the use of a domain equation involving a "relative" polymorphic product to construct a universe with the appropriate properties to serve as a polymorphic universe.

Angelic concurrency and its syntactic control

Ghica and Murawski (in FOSSACS'04 [11]) have introduced a (may-equivalence) fully abstract game model for a procedural language extended with primitives for parallel composition and synchronization on binary semaphores. The model uses an interleaved version of Hyland-Ong style games, where most of the original combinatorial constraints on plays are replaced by a simple principle naturally related to static process creation.

The game model of concurrency was then used (in ICALP'04 [13]) to show that 1st-order identifiers make program analysis undecidable. This led to the definition of an assume-and-guarantee proof system, expressed as a bounded type system called *Syntactic Control of Concurrency* (SCC) which can be analyzed via regular languages. Free identifiers in SCC are tagged with numbers that regulate the behaviour of the environment – they bound the number of concurrent threads that the identifiers range over. The type system can then be used to calculate the corresponding guarantees of the program.

Nominal games for nu-calculus

Many convenient features of modern programming languages involve some notion of *generativity*: the idea that an entity (e.g. reference, object, channel, etc.) may be freshly created, distinct from all others. The nu-calculus of Pitts and Stark was devised to explore this common property of generativity, by adding *names* to the simply-typed lambda-calculus. We have constructed (in LICS'04 [3]) the first fully abstract model for the nu-calculus using games constructed in the universe of FM-sets. In our setting, a play is a justified sequence of moves-with-names, satisfying certain conditions, but considered *up to appropriate renaming*. Intuitively the name set coupled with a move comprises all names that have been introduced by P at moves that are P-*visible* at that point. This device records the *scope* of each freshly created name. Name dependence of the various game constructions (such as plays, strategies, view functions, etc.) is *implicit*, and is achieved by the use of finite support.

Probabilistic IA with iteration

In a CONCUR'05 paper, Murawski and Ouaknine consider probabilistic 2nd-order IA programs with iteration and construct probabilistic automata that represent the game semantics. This yields a decision procedure for observational equivalence. Possible applications include verification of cryptographic protocols that use randomization to achieve anonymity, and verification of programs that use randomization as a means to improve performance (e.g. randomized quicksort). A more ambitious goal would be a model-checker that can compare randomized algorithms with their deterministic counterparts (e.g. primality tests).

Slot games and operational improvement

Slot games are essentially Hyland-Ong games augmented with a new action called token that represents a notion of resource consumption. Ghica (in POPL'05 [8]) has constructed a slot-game model for Concurrent Idealized Algol, and shown that it is fully abstract with respect to a notion of observation formalised in the operational theory of improvement of David Sands. Such a quantitative analysis of programs has many potential applications, ranging from compiler optimisations to resource-constrained execution and static performance profiling. The technique is illustrated with several examples that are known to be difficult to handle using known operational techniques.

Data abstraction refinement

Abstraction refinement has proved to be a highly effective method to verify systems with very large state spaces. Since abstractions are conservative over-approximations, safety of any abstracted program implies the safety of the concrete program (though the converse is not true. In a SAS'05 paper [8], Ghica *et al.* have introduced a purely semantic approach to (data) abstraction refinement, based on game semantics, using a language called Abstracted Idealized Algol. The key feature of the language is the use of abstraction schemes at the level of data-types, which allows the writing of abstracted programs in a syntax similar to that of concrete programs. In fact, a concrete program is a particular abstracted program, in which all the abstractions are identities. A fully abstract game model for AIA is presented, and an abstraction refinement semi-algorithm based on concrete representations of strategies is proposed.

Tool 1: Game semantics compiler for $\mathsf{IA}_2{+}\mathsf{while}$

To assess the practicability of the game-semantic approach to program verification, we have constructed a compiler that transforms an open procedural program into the finite-state machine representation of its fully abstract game semantics; very little user instrumentation of the source code is required. The tool was constructed in CAML; most of the back-end heavy duty finite-state machine processing was done using the AT+T FSM library. Ghica and Murawski's experiments (in TACAS'04 [2]) confirm what is a common situation in software model checking: even though the asymptotic complexity of the algorithm is high, the worst-case scenario only happens in pathological cases; many common and useful programs can in fact be verified. A number of case studies (bubble sort, abstract data type invariant, stack modules) have been carried out; further details can be found in a technical report [7].

Tool 2: Model-checker for higher-order SCC

Another tool constructed during the project is a model-checker for higher-order SCC programs using the process algebra CSP as an intermediate language. The tool can model standard concurrent protocols such as producer-consumer and has been used to verify equivalences between various mutual exclusion algorithms (Dekker's, Peterson's, Lamport's etc), certify abstract data type implementations and detect violations of safety properties. Our experiments (in TACAS'06 [12]) confirm that game semantics leads to much more compact models than those obtained by naïve interleaving (because the semantics captures extensional behaviour by hiding unobservable details such as internal state changes). We believe that it stands to gain further from partial-order reduction techniques, whose incorporation into game semantics is left for future work.

3 Research impact

The impact of our work can be seen in the invitations the project team has received to speak at international meetings, chair programme committees, lead research consortiums, and contribute to various publications.

Invited talks at meetings

1. North American School on Logic, Language and Information (NASLLI), Stanford, 24-30 June 2002 (Abramsky, 4 lectures)
2. Annual Meeting of the British Logic Colloquium, Birmingham, 12-14 September 2002 (Abramsky)
3. Symposium on Logic in Games and Multiagent Systems, Liverpool, 18-19 December 2002 (Abramsky)
4. Seminar on Logic and Informatics (SLI), Brussels, 31 March 2003 (Abramsky)
5. Int. Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS), Kanpur, 13-15 December 2002 (Ong)
6. Int. Conf. on Foundations of Software Science and Computation Structures (FOSSACS), Warsaw, 4-11 April 2003 (Abramsky)
7. Fields Institute Summer School on Logic and Foundations of Computation, Ottawa, 2-20 June 2003 (Abramsky, 6 lectures)
8. Fields Institute Workshop on Game Semantics, Ottawa, 2-20 June 2003 (Abramsky, Ghica, Murawski, Ong)
9. IMA Workshop on Agent Based Modeling and Simulation, Minneapolis, 3-6 November 2003 (Abramsky)
10. EU TMR Network of Excellence GAMES Annual Meeting, Vienna, 1-3 September 2003 (Ong)
11. Workshop of the Computational Metamodels project (CoMeta), Udine, 15-17 December 2003 (Abramsky)
12. Int. Symp. on Formal Methods for Components and Objects (FMCO), 2-5 November 2004 (Abramsky)
13. Workshop on the Mathematical Foundations of Programming Semantics (MFPS), Pittsburgh, 23-26 May 2004 (Ghica, Murawski, Ong)
14. ICCL Summer School on Proof Theory and Automated Theorem Proving, Dresden, 14-25 June 2004 (Ong, 5 lectures)
15. Workshop on Games in Design and Verification (GDV'04), Boston, 18 July 2004 (Ong)
16. 11th Workshop on Logic, Language, Information and Computation, Paris, 19-22 July 2004 (Ong)
17. Foundations of the Formal Sciences V: Infinite Games, Bonn, 26-29 November 2004 (Abramsky)
18. ETAPS'05 Workshop on Games for Logic and Prog. Lang., Edinburgh, 2-3 April 2005 (Ong)
19. Dagstuhl Seminar on Synthesis and Planning, Schloss Dagstuhl, 12-17 June 2005 (Ong)
20. 7th Int. LICS'05 Workshop on Logic and Computational Complexity, Chicago, 24-25 June 2005 (Ong)
21. EU Early Stage Research Training Network *Math. Logic and Applications*, 5-9 September 2005, Fischbachau, Germany (Ong, 4 lectures).
22. EU Research Training Network GAMES Spring School on Infinite Games and their Applications, Bonn, 15 - 19 March 2005 (Ong, lecture course)
23. Int. Static Analysis Symposium (SAS'05), London, 7-9 September 2005 (Abramsky)
24. Issac Newton Institute for Mathematical Science, Cambridge, *Logic and Algorithms*, 2-26 February 2006 (Ong, 5 lectures)
25. Geometry of Computation Workshop on Semantics and Games, Marseille, 20-24 February 2006 (Abramsky, Ghica, Murawski and Ong)
26. Midlands Graduate School on Foundations of Comp. , Leicester, 8-12 April 2006 (Ong, 4 lectures)

27. 34th Annual Spring School in Theo. Comp. Sc.: *Games: Between Semantics and Verification*, France, 29 May - 4 June 2006 (Abramsky, Ong)
28. 20th Annual Meeting of the European Association of Computer Science Logic (CSL 2006), Szeged, Hungary, 25-29 September 2006 (Ong)

Impact in influencing other research

Members of the project have edited special journal issues [5, 12] and contributed invited tutorials [1, 21] on game semantics. Recently Abramsky has developed a form of game semantics for multi-agent games, which can deal with partial information constructs such as the IF-quantifiers of Hintikka and Sandu. This was presented at the 7th Augustus de Morgan Workshop in London in November 2005, and a paper will appear in a Festschrift for Gabriel Sandu. Ong is principal organiser of the Issac Newton Institute Workshop on Games and Verification, 3-7 July 2006; he is an invited resident member of the Institute's six-month programme on Logic and Algorithm (January - June 2006). Ong has played a leading rôle in the EU Research Training Network *Games for Synthesis and Validation*. He is Programme Chair of the Network's final annual meeting, and co-leads a planning group to form a European research consortium on Games in Computer Science. Ong is a member of the Workshop Steering Committees of Games for Logic and Programming Languages, and Logic and Computational Complexity. Ong is Programme Chair of EATCS Conference *Computer Science Logic '05* and IEEE Symposium *Logic in Computer Science '07*.

Training and development of researchers

Both project postdocs have advanced their research programme and career considerably during the project. Soon after joining the project, Dan Ghica was awarded a Canadian National Science Research Council Postdoctoral Fellowship. He left the project in January 2005 to take up a Lectureship in Computer Science at the University of Birmingham. Andrzej Murawski was elected a Junior Research Fellowship of St. John's College, Oxford in the first year of the project. His FOSSACS'05 paper [15] won the EATCS Award for Best Theoretical Paper at ETAPS'05. Murawski now holds an EPSRC five-year Advanced Research Fellowship, and he is based in Oxford.

In addition, several doctoral students of Abramsky's and Ong's, who work on (algorithmic) game semantics, overlap with the project: Will Greenland on Game Semantics of Regions, William Blum on Game Semantics and Program Analysis, Sam Sanjabi on Game Semantics for Aspect Orientation, and Nikos Tzevelekos on Nominal Games and Objects. Several others (Jolie de Miranda, Matthew Hague and Yong Xie) work on topics in Games and Verification.

4 Project plan review

Targets in all three parts (Foundational, Application-oriented, and Implementation) of the Research Plan in the original proposal have virtually all been met. An unanticipated advance worth noting is the series of complexity characterizations that culminate in the complete classification [19] of the decidable fragments of IA.

The project was due to end on 6 April 2005, with about six man-months of postdoc salary left unspent[2]. We requested (and EPSRC approved) a six-month no-cost extension of the project, so that Murawski's employment could be extended by the same duration. The final six months proved to be especially productive: besides completing the pleasing classification result [19], Murawski struck up a new collaboration with Joel Ouaknine, resulting in a CONCUR'05 paper [20].

5 Explanation of expenditure

The expenditure in each of the three relevant categories (Staff, Travel and Subsistence, and Consumables) was largely in line with the plan in the proposal.

6 Further research & dissemination

The results of the project have laid the foundation for and inspired numerous further directions in Algorithmic Game Semantics. Four such projects currently enjoy EPSRC funding (awarded during the past year):

[2] The project was granted six postdoc years. Ghica began his postdoc two months after the project start date; he left four months before the end date to take up a Lectureship at Birmingham.

1. EP/C514645/1(P) *Pushdown Automata and Game Semantics*, £101,508 (Ong)
2. EP/D034906/1(P) *Modular Abstraction and Abstraction Refinement: A Game-Semantic Approach*, £128,758 (Ghica)
3. EP/C539753/1(P) *Towards a Game Semantics of Concurrency, Objects and Mobility* (Advanced Fellowship), £223,207 (Murawski)
4. EP/D037085/1 *Centre for Metacompuation* (Platform Grant) £431,197 (A., O., Melham, de Moor)

The Centre for Metacomputation will continue the themes of the project, while inter-twining them with the more applied work of colleages Tom Melham and Oege de Moor (and their groups), leading figures in hardware verification and programming tools respectively. We see this as an exciting opportunity for synergy between semantics and applications in verification and program analysis; it builds directly on the work which was done under this project.

Further dissemination activities include a number of invited lecture courses on Algorithmic Game Semantics to be given by the PIs over the next year (see items 24, 26 and 27 under "Invited talks" on p. 4).

References

[1] S. Abramsky. Algorithmic games semantics: a tutorial introduction. In H. Schwichtenberg and R. Steinbruggen, editors, *Proof and System Reliability*, pp. 21–47. Kluwer Academic Publishers, 2002.

[2] S. Abramsky, D. R. Ghica, A. S. Murawski, and C.-H. L. Ong. Applying game semantics to compositional software modelling and verification. In *Proc. TACAS'04*, pages 421–435, 2004. LNCS 2988.

[3] S. Abramsky, D. R. Ghica, A. S. Murawski, I. D. B. Stark and C.-H. L. Ong. Nominal games and full abstraction for the nu-calculus. In *Proc. LICS'04*, pages 150–159, 2004. IEEE Computer Society Press.

[4] S. Abramsky and R. Jagadeesan. A game semantics for generic polymorphism. In *Annals of Pure and Applied Logic*, 133: 3-37, 2005

[5] S. Abramsky and M. Mavronicolas (editors). *Game Theory Meets Theoretical Computer Science*. *TCS* vol. 343. 282 pages, 2005

[6] D. R. Ghica, A. Dimovski and R. Lazić. Data-Abstraction Refinement: A Game Semantic Approach. *Proc. SAS'05*. 2005. LNCS 3672.

[7] D. R. Ghica. Game-based Software Model Checking: Case Studies and Methodological Considerations. OUCL Tech. Report PRG-RR-03-11, May 2003

[8] D. R. Ghica. Slot games: a quantitative model of computation. In *Proc. POPL'05*, pp. 85–97. ACM Press, 2005.

[9] D. R. Ghica and G. McCusker. Reasoning about idealized algol using regular languages. In *Proc. ICALP'00*, pp. 103–116. 2000. LNCS 1853.

[10] D. R. Ghica and G. McCusker (editors). *Games for Logic and Programming languages. APAL (Special GaLoP'05 issue), Elsevier*. In progress.

[11] D. R. Ghica and A. S. Murawski. Angelic semantics of fine-grained concurrency. In *Proc. FOSSACS'04*, pp. 211–225. 2004. LNCS 2987.

[12] D. R. Ghica and A. S. Murawski. Compositional model extraction for higher-order concurrent programs. In *Proc. TACAS'06*, LNCS, 2006.

[13] D. R. Ghica, A. S. Murawski, and C.-H. L. Ong. Syntactic control of concurrency. *TCS*, 2006. Special ICALP'04 issue. To appear.

[14] A. Murawski. On program equivalence in languages with ground-type references. In *Proc. LICS'03*, pp. 108-117. IEEE Computer Society Press, 2003.

[15] A. Murawski and I. Walukiewicz. Third-order Idealized Algol with iteration is decidable. In *Proc. FOSSACS'05*, pp. 202–218. 2005. LNCS 3441.

[16] A. S. Murawski. About the undecidability of program equivalence in finitary languages with state. *ACM Transactions on Computational Logic*, 2003. Special LICS'03 issue. To appear.

[17] A. S. Murawski. Functions with local state: regularity and undecidability. *TCS*, 338(1/3):315–349, 2005.

[18] A. S. Murawski. Games for complexity of second-order call-by-name programs. *TCS*, 343(1/2):207–236, 2005.

[19] A. S. Murawski, C.-H. L. Ong, and I. Walukiewicz. Idealized Algol with ground recursion and DPDA equivalence. In *Proc. ICALP'05*, pp. 917–929. 2005. LNCS 3580

[20] A. S. Murawski and J. Ouaknine. On probabilistic program equivalence and refinement. In *Proc. CONCUR'05*, pp. 156–170. 2005. LNCS 3653

[21] C.-H. L. Ong. Model checking Algol-like languages using Game Semantics (invited paper). In *Proc. FSTTCS'02*, pp. 33–37. 2002. LNCS 2556.

[22] C.-H. L. Ong. An approach to deciding observational equivalence of Algol-like languages. In *APAL* (special LICS'02 issue), 130:125–171, 2004.