Volume 4, Number 3
July 2017

Published by the Association for Computing Machinery
Special Interest Group on Logic and Computation

SIGLOG news

TABLE OF CONTENTS

Association for Computing Machinery

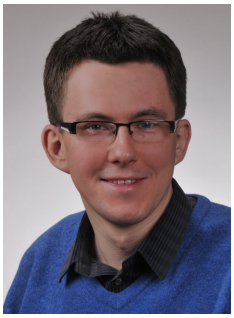
Advancing Computing as a Science & Profession

SIGLOG NEWS

Published by the ACM Special Interest Group on Logic and Computation

SIGLOG News (ISSN 2372-3491) is an electronic quarterly publication by the Association for Computing Machinery.

From the Editor



In this issue

– We announce the winners of this year's Alonzo Church, LICS Test-of-Time and Kleene Awards!

– Luc Segoufin surveys the current state of the art in first-order logics with tree-like models in Mikołaj Bojańczyk's column on Automata.

– Neil Immerman's Complexity column features an essay on thinking algorithmically about impossibility by Ryan Williams.

– Jean Krivine introduces us to Systems Biology in the Semantics column edited by Michael Mislove.

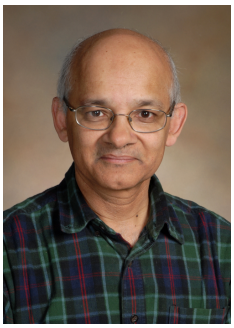
– Andrej Bauer reports on this year's MFPS and CALCO in the Conference Reports section, edited by Jorge A. Pérez.

– And, as usual, we wrap up with the latest issue of SIGLOG Monthly, prepared by Daniela Petrişan.

SIGLOG News is still looking for a volunteer to coordinate a section on book reviews. Please email `editor@siglog.org` if you are interested.

Andrzej Murawski
University of Warwick
SIGLOG News Editor

Chair's Letter



Summer is here! The conference season is underway. By the time this issue appears MFPS/CALCO (Ljubljana) and LICS (Reykjavik) will be over. I am only mentioning the conferences that I attended; there are many other meetings which is a great sign of the strength and vitality of the community. It is a great pleasure to announce that the second Church Prize was awarded to 6 people for work on game semantics. There is a longer announcement in this issue about it. The presentation is scheduled for CSL in Stockholm in August.

At LICS in Reykjavik, Valeria de Paiva made a passionate plea for greater diversity in the LICS community. I think it is fair to say that not everyone was sympathetic. I don't have the answers but certainly this issue cannot be ignored; it should be discussed widely. I think it would be a great idea if people were to write to me or to the Editor so we can publish some of the opinions in the following two or three issues.

Enjoy the rest of the summer!

Prakash Panangaden
McGill University
ACM SIGLOG Chair

AWARD ANNOUNCEMENTS

The 2017 Alonzo Church Award

Prakash Panangaden, School of Computer Science, McGill University

SIGLOG is delighted to announce that the 2017 Church Award goes to 6 people: Samson Abramsky, Martin Hyland, Radha Jagadeesan, Pasquale Malacaria, Hanno Nickau and Luke Ong for [Quoting from the official citation] "providing a fully-abstract semantics for higher-order computation through the introduction of games models, thereby fundamentally revolutionising the field of programming language semantics, and for the applied impact of these models."

These results appeared in three papers:

— S. Abramsky, R. Jagadeesan, and P. Malacaria. Full Abstraction for PCF. Information and Computation, Vol. 163, No. 2, pp. 409–470, 2000.
— J.M.E. Hyland and C.-H.L. Ong. On Full Abstraction for PCF: I, II, and III. Information and Computation, Vol. 163, No. 2, pp. 285–408, 2000.
— H. Nickau. Hereditarily sequential functionals. Proc. Symp. Logical Foundations of Computer Science: Logic at St. Petersburg (eds. A. Nerode and Yu.V. Matiyasevich), Lecture Notes in Computer Science, Vol. 813, pp. 253–264. Springer-Verlag, 1994.

The official citation gives a succinct summary of the contributions. The following paragraphs are taken from the official citation.

> These papers made two fundamental contributions to our understanding of programming languages and logic. First, they provided significant insight into the longstanding and fundamental "full abstraction problem" for the paradigmatic higher-order language PCF by giving a compositional semantic account of sequentiality, via an elegant cartesian-closed category of games and strategies. In the mid-1970s Milner posed the full-abstraction problem for PCF and Plotkin showed the difficulty of the problem, which essentially lies in the fact that the standard Scott-Strachey model permits non-sequential functions, although PCF itself is sequential.
>
> The papers constructed models for PCF from games, leading to the first fully abstract models of PCF whose construction made no reference to the syntax of PCF. The elements of the models are strategies for certain kinds of interactive dialogues between two players (or between system and environment). These dialogue games are required to follow certain conventions concerning when questions are posed or answered; these conventions reflect constraints on the information available to the players of the game. The papers give new insight into the fundamental work in higher-type recursion theory of such logicians as Kleene, Gandy, Normann, and Hyland.
>
> Second, and perhaps more importantly, game semantics has provided a new framework for the semantics of programming languages. Games can be used as a flexible and modular modelling tool, as a wide variety of programming language features can be understood as corresponding to different restrictions placed on allowed strategies. Thus there are fully abstract games models for call-by-value and call-by name languages; for languages with state, with control, with references, with exceptions, with nondeterminism, and

> with probability; and for concurrent and mobile process languages. In this way, game semantics has changed the landscape of programming language semantics by giving a unified view of the denotational universes of many different languages. This is a remarkable achievement that was not previously thought to be within reach. Techniques developed in game semantics have found their way into a wide variety of applications. For example, they have provided tools for the verification of a range of computational systems, such as reactive systems of timed and hybrid automata, higher-order and mobile processes, and model-checking higher-order programs. There has also been significant influence on other areas, for example, static program analysis, type systems, resource-sensitive compilation, and compiler certification. All of these developments can be traced from the initial work of Abramsky, Jagadeesan and Malacaria, of Hyland and Ong, and of Nickau.

In the following we expand on the above citation and give a few more details about the background.

The basic theories of Church and Turing give a model of how a computer program operates. The Church-Turing hypothesis places an ultimate limit on the power of computers. It states that any user requirement that cannot be coded as a program for a Turing machine can never, in fact, be implemented on a general-purpose computer. Furthermore, there is no more abstract way than an operational semantics of defining exactly which mathematical functions can be computed and which cannot.

The invention of denotational semantics was motivated by needing to reason about a program's behaviour and showing that it met its specification. Dana Scott [Scott 1970; Scott ; Scott 1976] brought to bear the well-known mathematical concepts of approximation and lattice theory to enable normal mathematical reasoning to apply to programming. He introduced the concept of continuity as a model-independent necessary condition for the computability of functions. Above all, denotational semantics emphasized compositionality as the key to attacking the problem of reasoning about complex programs.

There is a wide gap between the operational semantics of a programming language, which gives the "how" of programming, and the denotational semantics, which gives the "what." The problem of full abstraction is to explore, explain, and fill that gap. In particular, one seeks an abstract characterization of a necessary condition for those functions that are sequentially computed. It should be for sequentiality the same kind of model-independent condition as continuity is for computability.

The full abstraction problem was first articulated by Milner in his paper "Fully abstract models of typed lambda-calculi," which appeared in Theoretical Computer Science in 1977 [Milner 1977]. This work was prompted by Gordon Plotkin's discovery that the Scott model was not fully abstract because of the presence of functions like "parallel or" which were not ruled out in domain models but which could not be expressed in lambda calculus. This is not just an arcane technical point: one could not capture a fundamental property like sequentiality in the dominant Scott-Strachey paradigm of programming languages. Though Milner's journal article appeared before Plotkin's, his work was earlier and first appeared in a conference [Plotkin 1975].

Milner showed that one could construct such a model for the language PCF from the term model and that this model is unique. However this model was constructed from the syntax. Milner's construction raised the question whether there was a fully abstract model not derived from the syntax of the language. This was shown by Plotkin to be a hard problem in his paper, LCF Considered a Programming Language [Plotkin 1977] which also appeared in Theoretical Computer Science just a few months after Milner's paper. Plotkin showed that the usual domain theoretic models were not fully

abstract and that the problem lay in the fact that there was nothing in these models that rules out non sequential functions like “parallel or”. Berry and Curien’s [Berry and Curien 1982] model of sequential algorithms on concrete data structures had many of the important ingredients of the later games models.

There was one other major development in this period: the development of linear logic by Girard [Girard 1987]. This work, not motivated by full abstraction, but rather by a deep analysis of proof theory has illuminated and informed almost every aspect of research in semantics and type theory since its inception and played a key role in the full abstraction problem. Indeed several of the award winners [Hyland and Ong 1993; Abramsky and Jagadeesan 1994] were actively investigating games models of linear logic in the early 1990s following Blass’s construction of a games model for multiplicative linear logic [Blass 1992].

There were precursors to game semantics from higher-type recursion theory [Kleene 1959; Platek. 1966; Gandy 1967; Kleene 1978] where people were interested in whether there was a notion of sequential computation at higher type. In proof theory Lorenz and Lorenzen [Lorenzen 1960; Lorenz 1961; Lorenzen and Lorenz 1978; Lorenz 2001] pioneered the use of dialogue games to model provability. Unlike the game semantics for linear logic they were not trying to model proofs.

The papers for which the award is being given constructed fully abstract models for PCF from games. These were the first fully abstract models of PCF in which the construction made no reference to the syntax of PCF. The elements of the models are strategies for certain kinds of interactive “dialogues” between two players (or between “system” and “environment”). These dialogue games are required to satisfy certain conventions about when questions are posed or answered; these conditions in fact reflect constraints on the information available to the players of the game. For example, the *history-free* restriction [Abramsky et al. 2000] says that the play of the game in the past cannot affect the current move and Hyland and Ong [Hyland and Ong 2000] use a different restriction called *innocence* based on limiting the view of the past history. Independently Hanno Nickau [Nickau 1994] came up with a model essentially equivalent to the Hyland-Ong model. The conditions in question are *local* and *compositional*. It is remarkable that these restrictions on strategies turned out to be the key to full abstraction. One still has to construct an extensional collapse of the pure games model in order to obtain the fully abstract model.

It is, as far as I know, not known whether these two models are equivalent; Hyland and Ong conjectured that they were. The cartesian closed categories constructed in the two models are the same but the exact relation between history-freeness and innocence is far from clear.

More impressive than the PCF problem itself is the fact that the viewpoint of game semantics has been remarkably fruitful in capturing a number of other results. Indeed one can say without exaggeration that every major sequential programming paradigm has now been captured by imposing restrictions on the strategies that are allowed. Thus there are fully abstract games models for the language FPC [McCusker 1996], call-by-value languages [Honda and Yoshida 1997; Abramsky and McCusker 1998], for idealized Algol including state [Abramsky and McCusker 1997] and for languages with control [Laird 1997], references [Abramsky et al. 1998], exceptions [Laird 2001] nondeterminism [Harmer and McCusker 1999], concurrent and mobile processes [Laird 2005] and probability [Danos and Harmer 2000]. It is interesting that for PCF with exceptions the games model is extensional and fully abstract.

The fact that features, like mobility, that were not part of the original motivations of game semantics can be nicely handled, is another testament to the depth of these ideas. Furthermore, the treatment of all these features is remarkably modular in the sense that one can add or weaken appropriate restrictions on the games to pass from

one fully abstract model to another. The kind of overall view one has of the landscape of programming languages is a remarkable achievement that one would not have thought possible in, say, 1990. It is very clear from the fact that game models work so well for all these linguistic features that the semantics could not have been crafted from the syntax of one specific language like PCF.

As a testimony to the vigor of the subject we mention a topic that has grown significantly in the past decade. This is the subject of algorithmic game semantics. In a remarkable paper [Ghica and McCusker 2000] - which won a best paper award at ICALP - Dan Ghica and Guy McCusker captured the (game) semantics of a fragment of idealized Algol in a remarkably simple form, as regular expressions. This opens the way to using model checking and other automated techniques for reasoning about software correctness. This is a development with enormous potential impact on the practice of software verification. Intellectually, this sort of development helps bridge the gap between theory (semantics) and practice: language design, implementation and verification. This has been a very fruitful direction as we note below.

The subject of game semantics is an active and burgeoning area of research. In logic itself the original game-theoretic full completeness theorems continue to inspire new research by, among others, Abramsky and Melliès, and Hyland and his collaborators and students. In computer science the most pressing remaining questions relate to concurrency. For example, one seeks a semantic understanding of mobility in the sense of Milner's π-calculus [Milner 1999]. An illustration of one important breakthrough of game semantics is the introduction of two players. This permits a distinction between two kinds of non-determinism, the angelic kind (player's move) and the demonic kind (opponent's move).

From a purely mathematical point of view, it is possible to criticize game semantics as being too intensional. Rather than focusing on which functions are definable directly, the focus is on the computational processes by which these functions are computed. One has to quotient the space of these computational processes (games) in order to arrive at the fully abstract model. However, a remarkable result of Loader's [Loader 2001] shows that there is no effectively presented model for PCF, even when defined only over a finitary domain like the booleans (instead of the integers as is usual in PCF). Thus, while it is possible to give an extensional fully abstract model of PCF [O'Hearn and Riecke 1995], it is unlikely that one will find a model that has a simple presentation or gives the same insights into the nature of computational processes as games models do. Of course, it is still possible that there is an effective non-extensional model that is more abstract than the games model.

Games and game semantics have played a crucial role in the development of verification in areas beyond programming languages. Techniques developed in semantics have found their way into tools for the verification of a range of computational systems, including open probabilistic systems [Kiefer et al. 2012], reactive systems based on process calculi, such as CSP [Harmer and McCusker 1999; Dimovski and Lazic 2004; Abramsky 2010], CCS [Hirschowitz and Pous 2012], of timed and hybrid automata, as well as of computational models supporting higher-order and mobile processes, such as the pi-calculus [Eberhart et al. 2013]. While games and game semantics have their computer science roots in programming languages, the results devised there have found there way into these other, more abstract and more applied areas. All of these developments can be traced from the initial work of Abramsky, Jagadeesan and Malacaria and of Hyland and Ong and Nickau.

The uniform, precise representation of program behaviour by games led to notable applications in programming languages and new directions in verification and even implementation. Important papers include:

— Applications to static program analysis, as shown by extensive works by many authors [Ghica and McCusker 2000; Ghica 2005; Abramsky et al. 2004] and others.
— Higher-order model checking. Here game semantics is an essential ingredient to make higher-order computation concrete. The pioneering work is due to Ong [Ong 2006] with further important contributions by Naoki Kobayashi and others. Some key papers are [Neatherway et al. 2012] and [Kobayashi 2013].
— The contributions of the above work go beyond verification to the development of innovative type systems [Tsukada and Kobayashi 2012].
— Resource-sensitive compilation, pioneered by Dan R. Ghica. A hardware compiler based on game semantics and a type system for resource control appears in [Ghica and Smith 2011]. The use of game semantics for explicitly dealing with memory allocation is presented in [Fredriksson and Ghica 2013].
— Game semantics plays a key role in the certification of compositional compilers [Ramananandro et al. 2015; Stewart et al. 2015].

To summarize, these papers, through their study of a long-standing, fundamental problem, have changed the landscape of programming language semantics by giving a unified view of the semantic universe of many different languages and they have made an impact on the practice of programming languages through model checking, compilation and type systems. A primer on game semantics can be found in the April 2016 issue of SIGLOG News [Murawski and Tzevelekos 2016].

ACKNOWLEDGMENTS

This article was based on conversations with many people over the past several years. I would like to thank Samson Abramsky, Vincent Danos, Dan Ghica, Tony Hoare, the late Kohei Honda, Martin Hyland, Radha Jagadeesan, the late Robin Milner and Gordon Plotkin for their helpful comments.

Literatura

S. Abramsky. 2010. From CSP to Game Semantics. In *Reflections on the Work of C. A. R. Hoare*, C. B. Jones et al. (Ed.). Springer-Verlag, 33–45.

S. Abramsky, D. Ghica, A. Murawski, and L. Ong. 2004. Applying Game Semantics to Compositional Software Modelling and Verification. In *TACAS 2004: Tools and Algorithms for the Construction and Analysis of Systems: 10th International Conference (LNCS)*. Springer-Verlag, 421–435.

S. Abramsky, K. Honda, and G. McCusker. 1998. A fully abstract game semantics for general references. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*. IEEE Press, 334–344.

S. Abramsky and R. Jagadeesan. 1994. Games and Full Completeness for Multiplicative Linear logic. *Journal of Symbolic Logic* 59, 2 (1994), 543–574.

S. Abramsky, R. Jagadeesan, and P. Malacaria. 2000. Games and Full Abstraction for PCF. *Information and Computation* 163, 2 (2000), 409–470.

S. Abramsky and G. McCusker. 1997. Linearity, sharing and state: a fully abstract game semantics for Idealized Algol. In *Algol-like languages*. Vol. 2. Birkhauser, 297–329.

S. Abramsky and G. McCusker. 1998. Call-by-value games. In *Proceedings of CSL'97 (Lecture Notes In Computer Science)*. Springer-Verlag, 1–17.

G. Berry and P.-L. Curien. 1982. Sequential Algorithms on Concrete Data Structures. *Theoretical Computer Science* 20 (1982), 265–321.

A. Blass. 1992. A game semantics for Linear logic. *Annals of Pure and Applied Logic* 56 (1992), 183–220.

V. Danos and R. Harmer. 2000. Probabilistic Games Semantics. In *Proceedings of the Fifteenth IEEE Symposium On Logic In Computer Science*. IEEE Press, 204–213.

A. Dimovski and R. Lazic. 2004. Software model checking based on game semantics for CSP. In *Proceedings of Automated Verification of Critical Systems (ENTCS)*, Vol. 128. 105–125.

C. Eberhart, T. Hirschowitz, and T. Seiller. 2013. Fully abstract concurrent semantics for π-calculus. (2013). arXiv:1310.4306.

O. Fredriksson and D. R. Ghica. 2013. Abstract Machines for Game Semantics, Revisited. In *Proceedings of the ACM-IEEE Symposium On Logic In Computer Science*. 560–569.

R. O. Gandy. 1967. *Sets, Models and Recursion Theory*. North-Holland, Chapter Computable functionals of finite type I.

D. R. Ghica. 2005. Slot games: a quantitative model of computation. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages 2005 - POPL05*. ACM, 85–97.

D. R. Ghica and G. McCusker. 2000. Reasoning about idealized Algol using regular languages.. In *Proceedings of the 27th International Colloquium On Automata Languages And Programming, Geneva (Lecture Notes In Computer Science)*. Springer-Verlag, 103–116.

D. R. Ghica and A. I. Smith. 2011. Geometry of synthesis III: resource management through type inference. In *Proceedings of POPL 2011*. 345–356.

J.-Y. Girard. 1987. Linear Logic. *Theoretical Computer Science* 50 (1987), 1–102.

R. Harmer and G. McCusker. 1999. A fully abstract game semantics for finite nondeterminism. In *Proceedings of the Fourteenth IEEE Symposium On Logic In Computer Science*. IEEE Press.

T. Hirschowitz and D. Pous. 2012. Innocent strategies as presheaves and interactive equivalences for CCS. *Scientific Annals of Computer Science* 22 (2012).

K. Honda and N. Yoshida. 1997. Game-theoretic Analysis of Call-by-Value Computation. In *Proceedings of ICALP97 (Lecture Notes In Computer Science)*. 225–236.

J. M. E. Hyland and C.-H. L. Ong. 1993. Fair games and full completeness for Multiplicative Linear Logic without the MIX-rule. (1993). unpublished preprint.

J. M. E. Hyland and C.-H. L. Ong. 2000. On Full Abstraction for PCF I, II, III. *Information and Computation* 163, 2 (2000), 285–408.

S. Kiefer, A. S. Murawski, J. Ouaknine, B. Wachter, and J. B. Worrell. 2012. APEX: An Analyzer for Open Probabilistic Programs. In *Computer Aided Verification - 24th International Conference, CAV 2012, Berkeley, CA, USA, July 7-13, 2012 Proceedings*. 693–698.

S. C. Kleene. 1959. Recursive functionals and quantifiers of finite types I. *Trans. American Mathematical Society* 91 (1959), 1–52.

S. C. Kleene. 1978. Recursive functionals and quantifiers of finite types revisited. In *General Recursion Theory II, Proceedings of the 1977 Oslo Symposium*, J. E. Fenstad, R. O. Gandy, and G. E. Sacks (Eds.). North-Holland, 185–222.

N. Kobayashi. 2013. Model checking higher-order programs. *Journal of the ACM* 60, 3 (2013), 20.

J. Laird. 1997. Full abstraction for functional languages with control. In *Proceedings of the Twelfth IEEE Symposium On Logic In Computer Science*. IEEE Press, 58–67.

J. Laird. 2001. A fully abstract games semantics of local exceptions. In *Proceedings of the Sixteenth IEEE Symposium On Logic In Computer Science*. IEEE Press.

J. Laird. 2005. A Game Semantics of the asynchronous pi-calculus. In *Proceedings of the 16th International Conference on Concurrency Theory, CONCUR 2005. (Lecture Notes In Computer Science)*. 51–65.

R. Loader. 2001. Finitary PCF is not decidable. *Theoretical Computer Science* 266, 1-2 (2001), 341–366.

K. Lorenz. 1961. *Arithmetik und Logik als Spiele*. Christian-Albrechts-Universität zu Kiel.

K. Lorenz. 2001. Basic objectives of dialogue logic in historical perspective. *Synthese* 127, 1 (2001), 255–263.

P. Lorenzen. 1960. Logik und agon. In *Atti del XII Congresso Internazionale di Filosofia*, Vol. 4. 187–194.

P. Lorenzen and K. Lorenz. 1978. *Dialogische logik*. Wissenschaftliche Buchgesellschaft.

G. McCusker. 1996. Games and full abstraction for FPC. In *Proceedings of the Eleventh IEEE Symposium On Logic In Computer Science*. IEEE Press.

R. Milner. 1977. Fully abstract models of typed lambda-calculi. *Theoretical Computer Science* 4, 1 (1977), 1–23.

R. Milner. 1999. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press.

A. S. Murawski and N. Tzevelekos. 2016. An invitation to game semantics. *SIGLOG News* 3, 2 (2016), 56–67.

R. P. Neatherway, S. J. Ramsay, and C.-H. L. Ong. 2012. A traversal-based algorithm for higher-order model checking. In *ACM SIGPLAN International Conference on Functional Programming, ICFP'12, Copenhagen, Denmark, September 9-15, 2012*. 353–364.

H. Nickau. 1994. Hereditarily Sequential Functionals. In *Proc. Symp. Logical Foundations of Computer Science: Logic at St. Petersburg (Lecture Notes in Computer Science)*, Anil Nerode and Yu. V. Matiyasevich (Eds.), Vol. 813. Springer-Verlag, 253–264.

P. W. O'Hearn and J. G. Riecke. 1995. Kripke Logical Relations and PCF. *Information and Computation* 120, 1 (1995), 107–116.

C.-H. Luke Ong. 2006. On Model-Checking Trees Generated by Higher-Order Recursion Schemes. In *21th IEEE Symposium on Logic in Computer Science, 12-15 August 2006, Seattle, WA, USA, Proceedings*. 81–90.

R. A. Platek. 1966. *Foundations of Recursion Theory*. Ph.D. Dissertation. Standford University.

G. D. Plotkin. 1975. LCF as a programming language. In *Proceedings of the conference on Program Proving and Improving, Arc-et-Senans*.

G. D. Plotkin. 1977. LCF Considered a Programming Language. *Theoretical Computer Science* 5, 3 (1977), 223–256.

T. Ramananandro, Z. Shao, S-C. Weng, J. Koenig, and Y. Fu. 2015. A Compositional Semantics for Verified Separate Compilation and Linking. In *Certified Proofs and Programs*. 3–14.

D. Scott. Lattice Theory, Data Types and Semantics. In *NYU Symposia in Computer Science*, Randall Rustin (Ed.). 65–106.

D. Scott. 1970. *Outline of a mathematical theory of computation*. Technical Monograph PRG-2. Oxford University Computing Laboratory.

D. Scott. 1976. Data Types as Lattices. *SIAM Journal of Computing* 5, 3 (1976), 522–587.

G. Stewart, L. Beringer, S. Cuellar, and A. W. Appel. 2015. Compositional CompCert. In *Proceedings of the 42nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. ACM, 275–287.

T. Tsukada and N. Kobayashi. 2012. An Intersection Type System for Deterministic Pushdown Automata. In *Theoretical Computer Science - 7th IFIP Conference*. 357–371.

2017 LICS Test-of-Time Award

To acknowledge the long-term, foundational nature of papers appearing in LICS, the Test-of-Time Award is now awarded every year to the paper or papers that were published in LICS twenty years ago and that have "stood the test of time." The Test-of-Time Award Committee consisting of Christel Baier, Amy Felty (chair), Andrew Pitts and Nicole Schweikardt made two awards in 2017 to honor outstanding papers from the 12th IEEE Symposium on Logic in Computer Science, held 1997 in Warsaw, Poland:

Bisimulation for Labelled Markov Processes
by Richard Blute, Josée Desharnais, Abbas Edalat, Prakash Panangaden

> The paper introduces labelled Markov processes as a continuous-space variant of deterministic labeled transition systems where the dynamics of the state-action pairs is given by Markov kernels specifying the probability for measurable sets of successor states. The presented notion of bisimulation of labelled Markov processes is a conservative extension of Larsen and Skou's bisimulation for discrete probabilistic transition systems. The paper presents a highly non-trivial proof for the transitivity of bisimulation on labelled Markov processes and first steps towards a logical characterization of bisimulation in terms of a probabilistic Hennessy-Milner logic.
> By introducing labeled Markov processes and a notion of bisimulation for them, the authors provided important foundations for the formal semantics and analysis of stochastic systems where physical components interact with discrete ones. The paper opened a new research area on continuous-space stochastic models and inspired many researchers to study further properties of labelled Markov processes and variants thereof.

Towards a Mathematical Operational Semantics
by Daniele Turi, Gordon D. Plotkin

> This paper introduced a new and mathematically elegant way of relating the syntax and semantics of programs, using the existing category-theoretic notion of a distributive law between monads and comonads. Specifically, it gives an abstract view of the structural operational semantics of concurrent processes as distributing behaviour over syntax, one which guarantees the existence of a most abstract, compositional semantics of the language. The paper was an early example of the usefulness of coalgebraic techniques in semantics and has been, and still is, an extremely influential paper within the coalgebra community.

2017 Kleene Award

The LICS conference presents the Kleene Award annually to the best student paper or papers accepted for presentation at LICS that year. This year there were five eligible student papers accepted to LICS 2017. The program committee selected Amina Doumane for the 2017 Kleene Award for her paper "Constructive completeness for the linear-time mu-calculus".

Automata Column

MIKOŁAJ BOJAŃCZYK, University of Warsaw
bojan@mimuw.edu.pl

A central problem of logic in computer science is finding logics for which the satisfiability problem is decidable. A famous non-example is first-order logic, for which satisfiability is undecidable. One method to avoid such undecidability is to find syntactic restrictions on first-order logic (or even stronger logics, like second-order logic), which guarantee that satisfiable formulas have tree-like models, and then to use tree automata to check for the existence of tree-like models. Examples of logics covered by this method include modal logics and guarded logics. In this column, Luc Segoufin surveys the current state of first-order logics with tree-like models, with an emphasis on the newest member of the family, called guarded negation logic.

A survey on guarded negation

Luc Segoufin, INRIA and ENS-Cachan

We consider a logical framework building on existential positive formulas and then adding guarded negations and guarded fixpoints, where the guards are atomic formulas containing all free variables. The resulting first-order and fixpoint logics turn out to have nice algorithmic properties and nice expressive power. We survey some of them.

1. INTRODUCTION

First-order logic is widely used in mathematics, philosophy and computer science. It offers a good trade-off between simplicity and expressive power. In particular it is at the core of the relational database query language SQL.

However for certain applications it is too expressive. In particular it can describe the runs of a Turing Machine, making its satisfiability problem (asking whether a given formula is satisfied by at least one model) undecidable. This implies that first-order logic cannot be used for the formal methods approach in many areas of computer science, such as verification where satisfiability is crucial.

Several decidable fragments of first-order logic have been proposed. The most popular ones being the two-variable fragment [Mortimer 1975], and Modal Logic.

The case of Modal Logic is intriguing as many of its variations and extensions remain decidable (like adding two-way navigation, fixpoints etc.), while so many extensions of the two-variable fragment of first-order logic are undecidable [Grädel et al. 1999]. This led Moshe Vardi to wonder "why is modal logic so robustly decidable?" [Vardi 1996] and to answer by a combination of several key properties: (i) the tree model property (if a formula has a model then it has a model that is a tree) and (ii) translability into tree automata (each formula can be effectively translated into a tree automata recognizing its tree models). From the decidability of the emptiness problem of tree automata, these two properties already imply the decidability of the existence of a (possibly infinite) model. Finite satisfiability follows from a third key property: (iii) finite model property (if a formula has a model then it has a finite one). The first two properties are satisfied by all extensions of Modal Logic, in particular its two-way and fixpoint extensions. However, the finite model property is no longer satisfied by the fixpoint extension of Modal Logic, and if finite satisfiability is still decidable in this setting, it requires more complicated arguments [Bojańczyk 2003].

The situation is similar for guarded quantification first-order logic[1]. It was introduced in [Andréka et al. 1998] as an extension of Modal Logic with good algorithmic properties and again many of its extensions remain decidable, in particular the one with guarded fixpoints. As noticed by Erich Grädel [Grädel 2001], the reason is that it enjoys similar properties as Modal Logic. The tree model property is replaced by the tree-like model property: any satisfiable formula has a model of bounded tree-width.

[1] In order to avoid confusion with **guarded negation** logics, we use the terminology **guarded quantification** first-order logic instead of the usual one: guarded first-order logic.

The translability into tree automata should now be understood as transforming the formula into a tree automata recognizing the tree decompositions of those models of the formula that have bounded tree-width.

It turns out that the story continues. By moving the guards from quantifications to negations one gets a richer logic, guarded negation first-order logic, that has the tree-like model property and the translability to automata [Bárány et al. 2015]. This is the framework that we study in this survey.

We will see that guarded negation first-order logic, together with its extension with guarded fixpoints, has many nice properties. The first one being invariance under a notion of bisimulation that we survey in Section 3. This property, denoted guarded negation bisimulation, generalizes modal and guarded quantification bisimulation, yields the tree-like model property and opens the door to decidability.

Satisfiability for infinite models follows by translability into tree automata as explained in Section 4. For finite models, this is solved in the first-order case with the finite model property and for the fixpoint case this is achieved by a reduction to the modal case.

The nice algorithmic properties of guarded negation logic are further studied in Section 5, where the complexity of the model checking problem is studied.

Unlike the modal and the guarded quantification fragments, guarded negation first-order logic also has nice model theoretical properties. We will see in Section 6 that it has Craig Interpolation and the Projective Beth Property. These are the key to many applications, some of them being discussed in section 7.

2. PRELIMINARIES

Guarded negation logics are fragments of first-order logic and least fixpoint logic over relational schemes. Their models are relational structures. Before defining the logics, we recall some classical definitions.

Relational structures. A relational schema is a finite set of relation symbols, each having an associated arity. A relational *structure* M over a relational schema consists of a set, the *domain* of M, together with an interpretation of each relation symbol R of arity k of the schema as a k-ary relation over the domain denoted $R(M)$. A structure M is said to be *finite* if its domain is finite.

Homomorphisms. An *homomorphism* from a structure M to a structure N (assuming they have the same schema) is a mapping h from the domain of M to the domain of N such that for any relation symbol R of the schema and any tuple $\bar{a}$ that belongs to the interpretation of R in M, its image $h(\bar{a})$ belongs to the interpretation of R in N. This is not to be confused with the notion of *isomorphism* that requires moreover that the mapping is bijective and that the converse holds: $\bar{a}$ belongs to the interpretation of R in M iff $h(\bar{a})$ belongs to the interpretation of R in N. An homomorphism or an isomorphism is said to be *partial* if it is defined only on a subset of the domain of the structure.

Logic. We assume familiarity with first-order logic, FO, and least-fixpoint logic, LFP, over relational structures. We use classical syntax and semantics for FO and LFP. In particular we write $\phi(\bar{x})$ to denote the fact that the free variables of ϕ are exactly the variables in $\bar{x}$. We also write $M \models \phi(\bar{u})$ for the fact that the tuple $\bar{u}$ of elements of the model M makes the formula $\phi(\bar{x})$ true over M. A *sentence* is a formula with no free variable. It is either true or false over a structure and therefore defines a property of structures. The *size of a formula* ϕ is the number of symbols needed to write down the formula.

$(P)^*$	is	$P(x)$
$(\phi \wedge \psi)^*$	is	$\phi^*(x) \wedge \psi^*(x)$
$(\neg\phi)^*$	is	$\neg\phi^*(x)$
$(\langle R\rangle\phi)^*$	is	$\exists y\ R(x,y) \wedge \phi^*(y)$
$(\langle R^{-1}\rangle\phi)^*$	is	$\exists y\ R(y,x) \wedge \phi^*(y)$
$(\mathbf{S}\phi)^*$	is	$\exists y\ \phi^*(y)$

Fig. 1. Inductive translation of a modal logic formula ϕ to an equivalent UNFO-formula $\phi^*(x)$

This note surveys guarded negation logics. There is a first-order variant, that we now define. Later we will add fixpoints to it.

Guarded negation first-order logic is a fragment of first-order logic where any negated subformula must be *guarded* by an atom containing all its free variables.

Definition 2.1. The formulas of guarded negation first-order logic, denoted GNFO, are given by the following grammar, where R ranges over relation symbols of the schema and $\alpha(\bar{x}\bar{y})$ is an atomic formula:

$$\varphi ::= \quad R(\bar{x}) \mid x = y \mid \exists x\ \varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \alpha(\bar{x}\bar{y}) \wedge \neg\varphi(\bar{y}) \mid \neg\varphi(x) \mid \neg\varphi()$$

In words, any existential positive first-order formula, i.e. with no negation nor universal quantification, is a formula of GNFO. The syntax builds on existential positive first-order formulas by adding the negation of sentences, the negation of formulas with one free variable and formulas of the form $\alpha(\bar{x}\bar{y}) \wedge \neg\varphi(\bar{y})$, requiring that all the free variables $\bar{y}$ of φ occur in the atom $\alpha(\bar{x}\bar{y})$, called *the guard* of the negation.

If the formula only negates sentences or formulas with one free variable, then we say that it has *unary negations*. We denote by UNFO the set of unary negation first-order formulas. UNFO is a fragment of GNFO of independent interest. Unary negation can be seen as a special case of guarded negation, as $\neg\varphi(x)$ is equivalent to $x = x \wedge \neg\varphi(x)$, where the guard is an equality atom.

Notice that $x \neq y$ is not a formula of GNFO but $R(x,y,z) \wedge x \neq y$ is. Universal quantifications can only be used via double negations. For example $\forall\bar{x}\ \alpha(\bar{x}) \rightarrow \varphi(\bar{x})$ is equivalent to the GNFO formula $\neg(\exists\bar{x}\ \alpha(\bar{x}) \wedge \neg\varphi(\bar{x}))$.

Example 2.2. It is is easy to see that modal logic, and many of its extensions, is a fragment of GNFO, actually of UNFO. To see this consider the *global two-way modal logic*[2] defined by the grammar

$$\phi ::= P \mid \phi \wedge \phi \mid \neg\phi \mid \langle R\rangle\phi \mid \langle R^{-1}\rangle\phi \mid \mathbf{S}\phi$$

where P is a unary relation symbol (also called *proposition* in this setting), and R is a binary relation symbol (also called an *accessibility relation* in this context). Its semantics can be given via an inductive translation into UNFO as depicted in Figure 1.

Example 2.3. It turns out that GNFO also generalizes the *guarded quantification fragment of first-order logic*[3], denoted GFO, introduced by [Andréka et al. 1998] as a generalization of modal logic, requiring a guard on quantifications instead on negations. The logic GFO is the fragment of FO defined by the following grammar, where,

[2]Traditionally, the basic modal logic is defined without the backward and global features and can only navigate by traversing an edge in the forward direction.

[3]Recall that in order to avoid confusion with guarded **negation** logics, we refer to GFO as guarded **quantification** first-order logic, instead of its usual name: guarded first-order logic.

again, $\alpha(\bar{x}\bar{y}\bar{z})$ is an atomic formula (possibly an equality statement):

$$\varphi ::= R(\bar{x}) \mid x = y \mid \varphi \vee \varphi | \varphi \wedge \varphi | \neg\varphi \mid \exists \bar{x}\ \alpha(\bar{x}\bar{y}\bar{z}) \wedge \varphi(\bar{x}\bar{y}) \mid \forall \bar{x}\ \alpha(\bar{x}\bar{y}\bar{z}) \rightarrow \varphi(\bar{x}\bar{y})$$

By pushing the guards on the quantifications down to the atoms, it is easy to see that every GFO sentence is equivalent to a GNFO sentence. For instance the sentence $\exists x_1 x_2 x_3 E(x_1, x_2, x_3) \wedge E(x_1, y) \wedge \neg E(x_2, x_3)$ is equivalent to $\exists x_1 x_2 x_3 E(x_1, x_3) \wedge E(x_1, x_2, x_3) \wedge \neg E(x_2, x_3)$. This is only true for *sentences*, as $\neg R(xy)$ is a formula of GFO but is not expressible in GNFO. Guarded negation is strictly more expressive than guarded quantification as witnessed by the following sentence:

$$q = \exists xy\big(E(x, y) \wedge \neg(\exists uvw\ E(x, u) \wedge E(u, v) \wedge E(v, w) \wedge E(w, y))\big). \tag{1}$$

The formula q asks for the existence of an edge whose endpoints cannot be connected by a path of length 4. It is not equivalent to any GFO sentence as q defines a property that is not invariant under guarded quantification bisimulation, see Section 3.

GNFO has one important parametrization, denoted the *width*. Given a GNFO formula, one can push the existential quantifications up until a negation, a disjunction, or another existential quantification is reached. The maximal length of a block of existential quantifications in the resulting formula is called the width of the initial formula (see [Bárány et al. 2015] for a formal definition). We will see that the width of a formula is related to the tree-width of its models.

Guarded negation fixpoint logic is the extension of the guarded negation first-order logic with a guarded fixpoint modality. It is a syntactic fragment of least-fixpoint logic, from which it inherits the semantics.

Definition 2.4. For each relational schema τ, guarded negation fixpoint logic, denoted GNFP, is defined by the following grammar (notice that the first line correspond to the grammar of GNFO):

$$\begin{aligned}\phi\ ::=\ & R(\bar{x}) \mid x = y \mid \phi \wedge \phi \mid \phi \vee \phi \mid \exists x\, \phi \mid \alpha(\bar{x}\bar{y}) \wedge \neg\phi(\bar{x}) \mid \neg\varphi(x) \mid \neg\varphi() \mid \\ & \beta(\bar{u}\bar{w}) \wedge Z(\bar{u}) \mid \mu_{Z,\bar{z}}[\,\phi(\bar{Y}, Z, \bar{z})\,](\bar{x})\end{aligned}$$

where R is any relational symbol in τ, $\alpha(\bar{x}\bar{y})$ and $\beta(\bar{u}\bar{w})$ are atomic τ-formulas (possibly equality statements) and, in the last clause of the definition, the fixpoint variable Z occurs only positively in $\phi(\bar{Y}, Z, \bar{z})$, the ***matrix*** of the fixpoint, i.e. always under an even number of negations. For any value of $\bar{Y}$, the matrix formula can be seen as a function computing a relation, the valuations of $\bar{z}$ making it true, from the relation Z. The fact that Z occurs under an even number of negations ensures that this function is monotonic, and therefore that it has a least fixpoint, which is the semantics of the fixpoint formula.

Formulas of GNFP can be naturally thought of as being built up from atomic formulas using (i) guarded negation first-order formulas and (ii) guarded fixpoint operators. It is important to note that:

— no first-order parameters (i.e., free variables other than those $\bar{z}$ bound by the fixpoint operator) are permitted in the matrix of a fixpoint operator,
— free fixpoint variables $\bar{Y}$ other than Z are still allowed, enabling nesting and alternation of fixpoint definitions;
— fixpoint variables cannot be used as guards, and in fact, all atomic formulas involving fixpoint variables must be guarded by atomic τ-formulas or equalities.

As for the first-order case, we have a unary fixpoint variant denoted UNFP. It is the fragment of GNFP restricted to unary negations and unary fixpoints predicates. The width of a GNFP formula is defined as the width of its first-order parts.

Example 2.5. The fixpoint formula

$$\mu_{Z,x,y}[\, E(x,y) \vee \exists z\ (Z(x,z) \wedge E(z,y))\,](u,v)$$

computing the transitive closure of E is not a formula of GNFP as the matrix formula does not guard the variables x, z occurring in $Z(x,z)$.

The fixpoint formula

$$\mu_{Z,z}[\, y = z \vee \exists y'(Z(y') \wedge E(y',z))\,](x)$$

computing the connected component of y is also not a formula of GNFP as the matrix formula has y as a parameter.

However the fixpoint formula

$$\mu_{Z,z}[\, B(z) \vee \exists y'(Z(y') \wedge E(y',z))\,](x)$$

computing the set of nodes reachable from a node in B is in GNFP.

One could imagine different other syntaxes for guarding the fixpoint matrices in GNFP. For instance we could require that the whole matrix formula is guarded with a single atom or with a specific predicate signifying guardedness without expressly declaring any concrete guard. It turns out that this would not change the expressive power nor the complexity for satisfiability. However it does affect the succinctness of the formula and the complexity of the model checking. See the discussion concerning the syntax in [Bárány et al. 2015].

Example 2.6. Consider the following formula over a binary relation symbol E:

$$\mu_{X,y}[\, \neg\exists z(E(z,y) \wedge \neg X(z))\,]$$

It is a valid formula of UNFP as the monadic fixpoint variable X occurs within the scope of two negations. When evaluated on a directed graph, the first stage of the fixpoint computation contains all nodes y of in-degree 0. At any further stage of the fixpoint computation we add to X all points whose incoming nodes where all already in X. Hence if all the backwards paths starting from a given node are finite, this node will eventually be part of X. Conversely if a node has an infinite backward path, it will never be part of X.

Hence the GNFP formula:

$$\exists x\ \neg\mu_{X,y}[\, \neg\exists z(E(z,y) \wedge \neg X(z))\,](x) \tag{2}$$

expresses the fact that the graph contains an infinite backward path. In particular, in the case of finite structures, it expresses the fact that the graph contains a directed cycle.

Example 2.7. Greatest fixpoints can be simulated via the usual triple negations. It can be verified that these negations are legal (this follows from the fact that the matrix formula is guarded). For instance the formula of the previous example computes exactly the complement of the greatest fixpoint of the matrix formula $\exists z E(z,y) \wedge X(z)$, denoted $\neg\nu_{X,y}[\, \exists z E(z,y) \wedge X(z)\,]$.

Allowing simultaneous fixpoints in GNFP formulas does not increase the expressive power of the logic (although it can facilitate more succinct definitions).

Example 2.8. Guarded negation fixpoint logic generalizes the extension of the two-way modal logic of Example 2.2 with monadic fixpoint. It also generalizes the *guarded quantification fragment of fixpoint logic* (GFP) [Grädel and Walukiewicz 1999], in the sense that every sentence of GFP is equivalent to a sentence of GNFP.

Guarded quantification fixpoint logic is the fragment of least-fixpoint logic obtained by extending guarded quantification first-order logic with least fixpoints: given a formula $\phi(\bar{Y}, Z, \bar{z})$ that is positive in Z, has no free first-order variables other than $\bar{z}$ and Z has arity the number of variables in $\bar{z}$, the formula $\mu_{Z,\bar{z}}[\,\phi(\bar{Y}, Z, \bar{z})\,]$ is also a formula of GFP. Although the occurrences of fixpoint variables are not required to be guarded, in the context of a GFP *sentence*, every occurrence of an atom using a fixpoint relation is implicitly guarded, namely by the atom guarding the closest quantifier whose scope includes the occurrence in question). This implies that every sentence of GFP is equivalent to a sentence of GNFP, via a polynomial time transformation.

3. INVARIANCE BY BISIMULATION

The key to understand the expressive power of guarded negation logics, and also the key for its decidability, is the notion of *guarded negation bisimulation*. Bisimulation is a notion of similarity between two structures that is weaker than isomorphism. The desired property, Theorem 3.3 below, is that any two similar structures satisfy the same sentences of the logic.

This theorem has many important consequences. We describe two of them below.

The first one is a limitation on the expressive power of the logic. A property is said to be *closed under bisimulation* if any two similar structures either both satisfy the property or both falsify the property. It is important to understand that this notion has two flavors, depending on whether we consider only finite structures or all structures, finite or infinite. If we consider only finite structures, we say that a property is *closed under finite bisimulation* if any two similar *finite* structures either both satisfy the property or both falsify the property. Clearly closure under bisimulation implies closure under finite bisimulation. But the opposite is false. For instance if the property requires infinite models (this can be enforced in many ways, see for instance the paragraph after Theorem 4.3) then any two finite models falsify the property hence the property is trivially closed under finite bisimulation; however the property can additionally require another property that is not closed under bisimulation.

The key theorem mentioned above implies that a property that is not closed under bisimulation is not definable in the logic and a property that is not closed under finite bisimulation is not definable in the logic over finite models. We then say that the logic is *closed under bisimulation*.

The second consequence is the decidability of the satisfiability problem by showing that any structure is similar to a simple one, here of bounded tree-width, and then showing that it is decidable whether a formula has a simple model, see Section 4 below.

This is a classical track that has been successfully used for many logics. For instance modal logics, such as those defined in Section 2, define only properties closed under a notion of similarity known as *modal bisimulation*. Modal bisimulation has been extended to a notion of *guarded quantification bisimulation* that corresponds to the guarded quantification logics GFO and GFP.

We now introduce the appropriate notion of bisimulation for guarded negation logics, namely *guarded negation bisimulation* [4].

Let M be a relational structure. If a tuple of elements $\bar{a}$ from the domain of M belongs to the interpretation of a relation symbol R, then we say that $R(\bar{a})$ is a *fact* of M. We say that a tuple of elements of M is *guarded* if it is a singleton or there is a fact of M containing all its components. We denote by *guarded*(M) the set of guarded tuples of M. For a number k, we say that a tuple is k*-guarded* if it is guarded by a

[4]There is a stronger variant of guarded negation bisimulation introduced in [Bárány et al. 2013] that already characterizes GNFO.

fact of M using at most k elements of M. We denote by k-*guarded*(M) the set of all k-guarded tuples of M. In particular *guarded*$(M) = \bigcup_k k$-*guarded*(M).

Definition 3.1. Let M, N be two structures and k a strictly positive integer. A guarded negation bisimulation, GN-bisimulation in short, of width k is a non-empty binary relation $Z \subseteq k$-*guarded*$(M) \times k$-*guarded*(N) such that the following hold for every pair $(\bar{a}, \bar{b}) \in Z$, where $\bar{a} = a_1, \ldots, a_m$ and $\bar{b} = b_1, \ldots, b_n$

— the mapping sending $\bar{a}$ to $\bar{b}$ is a partial isomorphism (and in particular, $m = n$)
— [**Forward clause**] For every set X included in the domain of M and of size bounded by k there is a partial homomorphism $h : M \to N$ whose domain is X, such that h is consistent with the mapping sending $\bar{a}$ to $\bar{b}$ (i.e. $h(a_i) = b_i$ for all a_i in X), and such that for every $\bar{a}' \in k$-*guarded*(M) consisting of elements in X, the pair $(\bar{a}', h(\bar{a}'))$ belongs to Z.
— [**Backward clause**] For every set X included in the domain of N and of size bounded by k there is a partial homomorphism $h : N \to M$ whose domain is X, such that h is consistent with the mapping sending $\bar{b}$ to $\bar{a}$ (i.e. $h(b_i) = a_i$ for all b_i in X), and such that for every $\bar{a}' \in k$-*guarded*(N) consisting of elements in X, the pair $(h(\bar{a}'), \bar{a}')$ belongs to Z.

If we only require X to be finite and replace k-*guarded*(M) and k-*guarded*(N) by *guarded*(M) and *guarded*(N) in the forward and backward clauses, we then say that Z is a GN-bisimulation between M and N. In particular a GN-bisimulation is a GN-bisimulation of width k, for all k.

We write $M \approx_{GN} N$ if there is a GN-bisimulation between M and N and write $M \approx^k_{GN} N$ if there is a GN-bisimulation of width k.

Discussion. In the definition of guarded negation bisimulation, if the sets X are restricted to guarded tuples, instead of arbitrary finite sets, we then have the definition of guarded quantification bisimulation that was designed for guarded quantification logics [Flum et al. 2008].

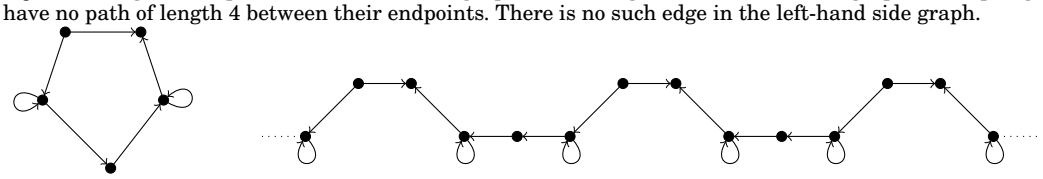
If we restrict in the definition of guarded negation bisimulation *guarded*(M) and *guarded*(N) to respectively the domain of M and the domain of N (in particular $Z \subseteq$ *dom*$(M) \times$ *dom*(N)) then we get the notion of *unary negation bisimulation* [ten Cate and Segoufin 2013] that is well suited for UNFO and UNFP as we shall see.

The notion of modal bisimulation, designed for two way modal logics, corresponds to the restriction of unary negation bisimulation, as defined previously, where in the forward and backward clauses for a pair (a, b) of Z, we only consider the sets X with two elements, the element a together with one element connected to a in the forward case, and similarly with b in the backward case.

This means that the existence of a guarded negation bisimulation implies the existence of a guarded quantification bisimulation which implies the existence of a modal bisimulation. This is to be expected as we have seen that modal logic is a fragment of guarded quantification logic which is also a fragment of guarded negation logic.

Example 3.2. Consider again the query q defined in GNFO by the formula (1), asking for the existence of an edge whose endpoints connot be connected by a path of length 4. This property is not closed under modal and guarded bisimulation. To see this consider the graph depicted in the left-hand side of Figure 2. The query is not satisfied by this graph: the top edge has its endpoints connected by a path of length 4 and any other edge starts or ends with a loop and therefore has its endpoints connected by a path of length 4. However this graph is guarded quantification bisimilar, and therefore also modal bisimilar, to the one depicted in the right-hand side of the figure, which makes the query true with any top edges. As the right-hand side graph is infinite, this

Fig. 2. Two guarded quantification bisimilar graphs. In the right-hand side infinite graph, the top edges have no path of length 4 between their endpoints. There is no such edge in the left-hand side graph.



implies that the property defined by q is not expressible in modal or guarded quantification logics over arbitrary graphs. However the right-hand side graph can be made finite by identifying top edges that are sufficiently far away in order to make sure that no loops of length 4 are created, hence still satisfying q. This implies that the property is also not definable over finite graphs.

The key property mentioned in the discussion above states that the existence of a guarded negation bisimulation implies indistinguishability by guarded negation fixpoint sentences. More precisely we have the following result:

THEOREM 3.3. *[Bárány et al. 2015]*

— *For every sentence* φ *of* GNFP *there is a* k *such that, if* $M \approx^k_{GN} N$ *then* M *and* N *agree on* φ.
— *If* $M \approx_{GN} N$ *then* M *and* N *satisfy the same sentences of* GNFP.

The number k associated to φ in Theorem 3.3 turns out to be its width as defined in Section 2. Hence any two structures that are GN-bisimilar with width k satisfy the same GNFP sentences of width k. A similar result can be shown for unary negation fragments:

THEOREM 3.4. *[ten Cate and Segoufin 2013]*

— *For every sentence* φ *of* UNFP *there is a* k *such that, if* $M \approx^k_{UN} N$ *then* M *and* N *agree on* φ.
— *If* $M \approx_{UN} N$ *then* M *and* N *satisfy the same sentences of* UNFP.

We now argue that guarded negation bisimulation is the best possible notion of similarity under which guarded negation logics are invariant. Indeed, guarded negation bisimulation invariance can be used to *characterize* GNFO: any property that is definable in first-order logic and closed under guarded negation bisimulation is actually definable in GNFO. This result works for both types of closure, closure under finite guarded negation bisimulation and closure under arbitrary guarded negation bisimulation. Similar results were obtained earlier showing that modal bisimulation characterizes modal logic [van Benthem 1983; Rosen 1997] and that guarded quantification bisimulation characterizes guarded quantification first-order logic [Andréka et al. 1998; Otto 2010].

THEOREM 3.5.

— *a first-order property is definable in* GNFO *iff it is closed under GN-bisimulation.*
— *a first-order property is definable in* GNFO *over finite structures iff it is closed under finite GN-bisimulation.*

The characterization of GNFO was obtained in [Bárány et al. 2015] for the infinite case and in [Otto 2013] for the (considerably harder) finite case.

Similar results were obtained for UNFO.

THEOREM 3.6. *[ten Cate and Segoufin 2013]*

— *a first-order property is definable in* UNFO *iff it is closed under UN-bisimulation.*
— *a first-order property is definable in* UNFO *over finite structures iff it is closed under finite UN-bisimulation.*

The results of Theorem 3.5 and Theorem 3.6 can actually be refined further by showing in each setting, that for each k, a first-order property is closed under GN-bisimulation of width k iff it is definable by a GNFO sentence of width k (and similarly for UN-bisimulation and UNFO).

The situation for fixpoint logics is not yet entirely solved. It is known that the μ-calculus is the modal bisimulation invariant fragment of monadic second order logic (MSO) [Janin and Walukiewicz 1996] and that the guarded quantification fixpoint logic is the guarded quantification bisimulation invariant fragment of guarded second order logic (GSO, where second order quantification is limited to relations containing guarded tuples) [Grädel et al. 2002]. However those results only hold for infinite structures. We don't know yet whether they are true over finite structures. For guarded negation fixpoint logic, only the unary negation case has been partially answered. It is known that the sentences of UNFP of width k are precisely the properties expressible in GSO and invariant under unary negation bisimulations of width k [Benedikt et al. 2015].

4. DECIDABILITY

In this section we consider the satisfiability problem. For this problem the input is a sentence of the logic and the question is whether this sentence has a model, i.e. a structure making the sentence true. This problem has two variants depending on whether we ask for a *finite* model or an arbitrary one. It is usually easier to decide whether a sentence has a model while it is harder to decide whether it has a finite model. In some cases, both the finite and infinite satisfiability are equivalent in the sense that a sentence has a model iff it has a finite model. We then say that the logic has the *finite model property*.

In general the satisfiability problem for first-order logic is undecidable, both in its finite or infinite variants. However several fragments of first-order logic are decidable. This is in particular the case for the frameworks already mentioned in this note: for instance it is decidable whether a sentence of the μ-calculus, with its two-way extension, has a model. The problem is ExpTime-complete both for infinite satisfiability [Vardi 1998] and for finite satisfiability [Bojańczyk 2003]. For guarded quantification first-order logic, GFO, satisfiability is 2ExpTime-complete [Grädel 2001]. Because GFO has the finite model property, it does not make any difference whether we consider finite or infinite satisfiability. Note that the complexity lowers to ExpTime-complete if the maximal arity of the relations in the schema is fixed (for instance graphs). For guarded quantification fixpoint logic, GFP, the satisfiability problem is ExpTime-complete [Grädel and Walukiewicz 1999]. Finite satisfiability was considerably harder to achieve but turns out to be also decidable within the same complexity bounds [Bárány and Bojańczyk 2012].

An important consequence of GN-bisimulation, actually GN-bisimulation of width k for some k, is decidability. It is easy to see that any structure is GN-bisimilar of width k to a structure of tree-width k. It is not important to know the definition of tree-width to understand this note. A structure of tree-width 1 is a tree and the smaller the tree-width is the more the structure resembles a tree. The interested reader is refered to [Flum et al. 2008] for more details. The important result relating tree-width and decidability is Courcelle's Theorem stating that, for any k, it is decidable whether a sentence of monadic second-order logic has a model of tree-width k [Courcelle 1990].

A GN-bisimulation can be seen as a game between two players. A configuration of the game consists of a k-guarded tuple $\bar{a}$ of structure M and a k-guarded tuple $\bar{b}$ of structure N. In a round of the game the first player chooses a set X of size at most k in either M or N. The second player must respond with a partial homomorphism h from X to the other structure. Finally the first player chooses a k-guarded tuple $\bar{a}'$ within X and the game resumes with the pair $\bar{a}'$ and $h(\bar{a}')$. It is now easy to verify that the existence of a GN-bisimulation is equivalent to the fact that the second player has a strategy for playing this game forever. This strategy can be encoded as a tree structure where every node corresponds to the current configuration and the edges to its children correspond to the answers of the second player to each possible move of the first player. The resulting tree is infinite as the play goes on forever.

Consider any structure M. There is a trivial GN-bisimulation between M and itself, hence a trivial strategy for the second player to play forever. As explained above, this strategy yields an infinite tree structure that can be seen as a tree decomposition of a new (infinite) structure N, witnessing that N has tree-width k and that M is GN-bisimilar to a structure of tree-width k. The structure N obtained this way is called the *GN-unraveling of M of width k*.

Hence, by invariance under GN-bisimulation, if a sentence φ of GNFP has a model then there is a k such that φ has a model of tree-width k. This is known as the *tree-like model property*.

THEOREM 4.1. *[Bárány et al. 2015]* GNFP *has the tree-like model property.*

Theorem 4.1 can be used to decide whether a sentence of GNFP has a model. For this, one constructs from a sentence φ of GNFP a MSO formula recognizing the unravelingsof the models of φ of width k, where k is the width of φ (the width is computable from φ). This implies decidability using Courcelle's Theorem. However this strategy does not give a good complexity bound. In order to obtain the optimal complexity bound given below one needs to directly compute a small automaton working on the tree decompositions of the structures of width k, and this requires more work.

THEOREM 4.2. *[Bárány et al. 2015] It is 2*EXPTIME*-complete to decide whether a sentence of* GNFP *is satisfiable.*

Note that if the upper bound applies to GNFP the lower bound already holds for UNFO [ten Cate and Segoufin 2013] and already in the case of graphs.

What about finite satisfiability? How can we decide whether a sentence has a finite model? We cannot use anymore the tree-like model property for finite satisfiability as the unravelings are inherently infinite.

Like for modal and guarded quantification logics, the situation is then different depending on whether we consider the first-order case or the fixpoint case. The simplest of the two is the case of GNFO. In this case one can show that the logic has the finite model property: if a sentence has a model it has a finite one. Hence finite satisfiability is equivalent to satisfiability and Theorem 4.2 applies. More precisely we can show that GNFO sentences have a small model property:

THEOREM 4.3. *[Bárány et al. 2015] Every satisfiable sentence φ of* GNFO *has a finite model of size double exponential in the size of φ.*

The finite model property does not hold for GNFP. To see this recall the formula (2) of Example 2.6 expressing the fact that a graph has an infinite backward path: $\exists x\ \neg\mu_{X,y}[\ \neg\exists z(E(z,y) \wedge \neg X(z))\ ](x)$.

Consider now the following GNFP formula:

$$\exists x\ \neg\exists y\ E(x,y) \vee \exists x \neg\mu_{X,y}[\ \neg\exists z(E(z,y) \wedge \neg X(z))](x)$$

expressing the property that either there exists a maximal element or there is an infinite backward path. This formula is obviously false in the infinite structure $(\mathbb{N}, suc)$. However it holds on any finite structure as if a finite structure has no maximal elements, it must contain a cycle, and hence an infinite backward path. The negation of this sentence is satisfiable, by $(\mathbb{N}, suc)$, but has no finite model.

The decidability for finite satisfiability turns out to be considerably more difficult. It has been achieved by reducing the finite satisfiability problem for GNFP to the finite satisfiability problem for GFP. The latter reduces to the finite satisfiability problem for the two-way μ-calculus [Bárány and Bojańczyk 2012], whose finite satisfiability was obtained in [Bojańczyk 2003]. Altogether it can be achieved within the same complexity bounds as for the infinite case.

THEOREM 4.4. *[Bárány et al. 2015] It is 2*EXPTIME*-complete to decide whether a sentence of* GNFP *has a a finite model.*

5. MODEL CHECKING

In this section we study the model checking problem for guarded negation logics. In the model checking problem, the input consists of a sentence and a structure and the goal is to decide whether the structure satisfies the sentence.

In the case of modal logics and guarded quantification logics, the model checking for GFO is PTime-complete and the one of GFP is the same as for the μ-calculus and lies between PTime and NP $\cap$ coNP [Berwanger and Grädel 2001]. It is actually a famous open problem to know whether there exists a polynomial time algorithm for the model checking problem of μ-calculus. Equivalently this amounts to find a polynomial time algorithm for solving parity games (see [Flum et al. 2008]).

Before stating the results for guarded negation logics we briefly review the complexity classes involved.

The first class we use is denoted P^{NP}, also known as Δ_2^p. It consists of all problems that are computable by a Turing machine running in time polynomial in the size of its input, where the Turing machine, at any point during its computation, can ask yes/no queries to an NP oracle, and take the answers of the oracle into account in subsequent steps of the computation (including subsequent queries to the NP oracle). Analogously, one can define the classes $\mathrm{NP}^{\mathrm{NP}}$ and $\mathrm{coNP}^{\mathrm{NP}}$, which are also known as Σ_2^p and Π_2^p, respectively. An example of a P^{NP}-complete problem is LEX(SAT), which takes as input a Boolean formula $\phi(x_1, \ldots x_n)$ and asks what is the value of x_n in the lexicographically maximal solution (where x_n is treated as the least significant bit in the ordering) [Wagner 1987].

Inside P^{NP} lies a hierarchy of classes $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$ with $i \geq 1$. They are defined in the same way as P^{NP}, except that the number of yes/no queries that can be asked to the NP oracle is bounded by $O(\log^i(n))$, where n is the size of the input. An example of complete problem for $\mathrm{P}^{\mathrm{NP}[O(\log^i n)]}$ is the problem LEX_i(SAT) testing, given a Boolean formula $\phi(x_1, \ldots, x_n)$ and a number $k \leq \log^i(n)$, whether the value of x_k is 1 in the lexicographically maximal solution [ten Cate and Segoufin 2013].

We are now ready to state the model checking results for guarded negation logics.

THEOREM 5.1. *[Bárány et al. 2015] The model checking problem for* GNFO *is* $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$*-complete under polynomial time reductions. For* GNFP *it is in* $\mathrm{NP}^{\mathrm{NP}} \cap \mathrm{coNP}^{\mathrm{NP}}$ *and hard for* P^{NP}.

The hardness results already hold for unary negated formulas.

The model checking problem for GNFO provides one of the few natural complete problems for the complexity class $\mathrm{P}^{\mathrm{NP}[O(\log^2 n)]}$.

For UNFP, the gap between the upper bound and the lower bound reflects the similar open problem for GFP and the μ-calculus where the model checking problem lies between PTime and NP $\cap$ coNP [Berwanger and Grädel 2001].

The result for GNFP is sensitive to the syntax. Various ways of guarding the fixpoint leads to different complexities. For instance it becomes ExpTime-complete if we use a syntactic clause "guard" whose semantics contains all guarded tuples [Bárány et al. 2015].

6. MODEL THEORY

A nice thing with guarded negation first-order logic is that it also behaves well in terms of model theoretic properties. In particular it has Craig-Interpolation, and therefore the Projective Beth Property. For the guarded quantification logic GFO, only Beth Property holds [Hoogland et al. 1999]. Both Craig-Interpolation and Projective Beth Property fail for GFO [Bárány et al. 2013].

Given two formulas φ and ψ, we write $\varphi \models \psi$ if any model of φ is also a model of ψ, in other words, if $\varphi \wedge \neg\psi$ is not satisfiable. A logic has Craig-Interpolation if whenever two formulas φ and ψ of the logic are such that $\varphi \models \psi$, then there is a third formula θ of the logic, using only the relation symbols occurring in both φ and ψ such that $\varphi \models \theta$ and $\theta \models \psi$.

In particular, consider a formula $\varphi(R)$ expressing a property of a relation R and such that $\exists R\ \varphi(R) \models \forall R\ \varphi(R)$. We then say that φ is R-invariant (the result does not depend on the actual value of R). Let $\varphi(S)$ be the formula constructed from $\varphi(R)$ replacing the symbols R by a fresh new symbol S. It follows from $\exists R\ \varphi(R) \models \forall R\ \varphi(R)$ that $\varphi(R) \models \varphi(S)$. As R and S do not appear in both sides, Craig-Interpolation implies in this case that there exists a formula θ, which does not mention R nor S and such that θ is equivalent to $\exists R\ \varphi(R)$. In the literature we then say that R-invariant properties are expressible without R.

Another consequence of Craig-Interpolation is the Projective Beth Property.

Let σ, τ be two relational schemes such that $\sigma \subset \tau$ and R is a relation symbol in $\tau \setminus \sigma$. [5] Let φ be a formula over the schema τ. We say that φ *implicitly defines* R if for any structure M over σ and any two τ-expansions[6] M_1 and M_2 of M such that $M_1 \models \varphi$ and $M_2 \models \varphi$ then we have $R(M_1) = R(M_2)$. In other words the formula φ defines a partial function associating to a model M over σ an instantiation for R, namely $R(M_1)$ for an arbitrary τ-expansion of M, the definition ensuring that it does not depend on the choice of M_1. The mapping is partial because M may not have a τ-expansion satisfying φ.

Example 6.1. Consider the schema $\sigma = \{E\}$ where E is binary whose models are directed graphs. Consider the extension τ of σ by four unary predicates R, P_0, P_1, and P_2.

The following formula says that nodes in R must be on a cycle of length 4.

$$\forall x\ R(x) \rightarrow \exists u, v, w\ E(x,u) \wedge E(u,v) \wedge E(v,w) \wedge E(w,x)$$

[5] Beth Property requires $\tau = \sigma \cup \{R\}$. The Projective Beth Property is a generalization of Beth Property where τ (and therefore the formula) may contain more relations

[6] A τ-expansion of M is a structure over the same domain as M, with the same interpretation of relation symbols in σ. Hence the τ-expansions of M only differ by their interpretations of the relation symbols in $\tau \setminus \sigma$

It does not implicitly define R as these nodes may or may not be in R. However, in conjunction with the following formula, implying that when a node is not in R then it cannot be on a cycle of length 4,

$$\forall x \ \neg R(x) \rightarrow P_0(x) \wedge \neg P_1(x) \wedge \neg P_2(x)$$
$$\forall x, y \ P_i(x) \wedge E(x,y) \rightarrow P_{i+1 \ mod \ 3}(y)$$

the resulting formula implicitly defines R as the nodes lying on a cycle of length 4. Note that all formulas are in UNFO as all negations are unary. Note that the mapping is partial as in some graphs, it is not possible to assign colors such that the whole formula is true. But when this is possible, R contains all nodes in a cycle of length 4.

A logic has the Projective Beth Property if every relation implicitly definable in the logic by a formula φ over the schema τ has an explicit definition in the logic, i.e. there is a formula $\phi(\bar{x})$ over the schema σ (in particular ϕ does not use the symbol R) such that $\varphi \models \forall \bar{x} \ R(\bar{x}) \leftrightarrow \phi(\bar{x})$.

Example 6.2. For instance the formula of Example 6.1 implicitly defining the nodes sitting on a cycle of length 4 in a graph can be explicitly defined by the formula

$$\exists u, v, w \ E(x,u) \wedge E(u,v) \wedge E(v,w) \wedge E(w,x).$$

THEOREM 6.3. GNFO *has Craig-Interpolation and therefore also the Projective Beth Property.*

Theorem 6.3 was first proved in [Bárány et al. 2013]. A constructive proof was then given in [Benedikt et al. 2016]. Actually GNFO even has Lyndon Interpolation (the interpolant θ also preserves the positivity of the relations symbols) [Benedikt et al. 2016].

When both sides of $\varphi \models \psi$ have only unary negations, the interpolant can be chosen in UNFO [ten Cate and Segoufin 2013]. One can further assume that the interpolant uses the same number of variables as φ and ψ [ten Cate and Segoufin 2015].

There are very few results of this kind for fixpoint logics. However Craig Interpolation does hold for unary negation fixpoint logic, UNFP, but fails for guarded negation fixpoint logic, GNFP [Benedikt et al. 2015].

7. APPLICATIONS

Databases. There are numerous applications in databases. A *view* of a database is a new database providing a subset of its information possibly restructured in a different way. A view is usually specified using queries, each query defining a new relation of the view. Views have many applications, a notable one being privacy, when some users may not be entitled to see all of the database. In this context it may be useful to known whether a view specification leaks some important confidential information. The problem of view determinacy models this by asking whether a view specification contains enough information for answering a given target query [Nash et al. 2010]. This problem is known to be already undecidable for views specified by means of conjunctive queries and a conjunctive target query [Gogacz and Marcinkowski 2016]. However for view specifications defined by answer-guarded[7] GNFO queries and the target query is also answer-guarded the problem becomes decidable. It is a consequence of the Projective Beth Property of GNFO that if a view determines a query then a rewriting of the query can be found in GNFO. Decidability follows from the decidability of GNFO [Bárány et al. 2013].

[7]A query is answer guarded if it is of the form $\alpha(\bar{x}) \wedge \varphi(\bar{x})$ where α is an atom.

Database systems are constantly facing incomplete or inconsistent data. In this case, computing the answers to a query requires some reasoning and therefore a decidable logic. Several decidable scenarios with integrity constraints specified using guarded negation rules where considered in [Bárány et al. 2013; Bourhis et al. 2016; Bourhis et al. 2014]. See also [Bárány et al. 2013; Bienvenu et al. 2014].

Trees. When restricted to unranked ordered tree structures (also known as XML trees), UNFO has the same expressive power as Core-XPath, the navigational fragment of XPath [ten Cate and Segoufin 2013]. This equivalence of expressive power holds when Core-XPath is viewed as a language of sentences (existence of a path) and then it corresponds to UNFO sentences. The equivalence also holds when Core-XPath is viewed as a language defining properties of a node (existence of a path starting from this node) and then it corresponds to UNFO formulas with one free variable. In this case it also has the same expressive power as FO^2 [Marx and de Rijke 2005]. The equivalence also holds when Core-XPath is viewed as a language defining pairs of nodes (with a certain path linking them) and then it corresponds to UNFO formulas with two free variables. In this setting UNFP sentences define precisely the regular tree languages.

Boundedness. In some cases it is useful to know whether a property expressed using a fixpoint formula does require a fixpoint. In other words whether the fixpoint computation can be replaced by a first-order formula. This is known as the *boundedness problem*. This problem is decidable for a fragment of guarded negation fixpoint: guarded negation Datalog [Bárány et al. 2012].

8. CONCLUSION

We have seen that guarded negation logics have nice algorithmic properties and nice expressive power.

There exists several decidable extensions of what we presented in this paper. For instance atomic guards can be replaced by "clique guards" [Bárány et al. 2015]. However a little bit of unguarded negation, like adding just inequality, kills decidability [Bárány et al. 2015]. In the fixpoint case it is possible to relax the requirement forbidding the existence of parameters. Satisfiability of an infinite model is then decidable, the finite case being open [Benedikt et al. 2016].

REFERENCES

Hajnal Andréka, Johan van Benthem, and István Németi. 1998. Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic* 27 (1998), 217–274.

Vince Bárány, Michael Benedikt, and Pierre Bourhis. 2013. Access patterns and integrity constraints revisited. In *Intl. Conf. on Database Theory (ICDT)*.

Vince Bárány, Michael Benedikt, and Balder ten Cate. 2013. Rewriting Guarded Negation Queries. In *Intl. Symp. on Mathematical Foundations of Computer Science (MFCS)*.

Vince Bárány and Mikołaj Bojańczyk. 2012. Finite satisfiability for guarded fixpoint logic. *Inform. Process. Lett.* 112, 10 (2012), 371–375.

Vince Bárány, Balder ten Cate, and Martin Otto. 2012. Queries with Guarded Negation. In *Intl. Conf. on Very Large Databases (VLDB)*.

Vince Bárány, Balder ten Cate, and Luc Segoufin. 2015. Guarded Negation. *J. ACM* 62, 3 (2015), 22:1–22:26.

Michael Benedikt, Pierre Bourhis, and Michael Vanden Boom. 2016. A Step Up in Expressiveness of Decidable Fixpoint Logics. In *Symp. on Logic in Computer Science (LICS)*. 817–826.

Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. 2015. Interpolation with Decidable Fixpoint Logics. In *Symp. on Logic in Computer Science (LICS)*. 378–389.

Michael Benedikt, Balder ten Cate, and Michael Vanden Boom. 2016. Effective Interpolation and Preservation in Guarded Logics. *ACM Trans. Computational Logic* 17, 2 (2016), 8:1–8:46.

Dietmar Berwanger and Erich Grädel. 2001. Games and Model Checking for Guarded Logics. In *Logic for Programming, Artificial Intelligence, and Reasoning (LPAR)*. 70–84.

Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. 2014. Ontology-Based Data Access: A Study through Disjunctive Datalog, CSP, and MMSNP. *ACM Trans. Database Systems* 39, 4 (2014), 33:1–33:44.

Mikołaj Bojańczyk. 2003. The finite graph problem for two-way alternating automata. *Theoretical Computer Science* 3, 298 (2003), 511–528.

Pierre Bourhis, Marco Manna, Michael Morak, and Andreas Pieris. 2016. Guarded-Based Disjunctive Tuple-Generating Dependencies. *ACM Trans. Database Systems* 41, 4 (2016), 27:1–27:45.

Pierre Bourhis, Michael Morak, and Andreas Pieris. 2014. Acyclic Query Answering under Guarded Disjunctive Existential Rules and Consequences to DLs. In *Intl. Workshop on Description Logics*.

Bruno Courcelle. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Information and Computation* 85, 1 (1990), 12–75.

Jörg Flum, Erich Grädel, and Thomas Wilke (Eds.). 2008. *Logic and Automata: History and Perspectives [in Honor of Wolfgang Thomas]*. Texts in Logic and Games, Vol. 2. Amsterdam University Press.

Tomasz Gogacz and Jerzy Marcinkowski. 2016. Red Spider Meets a Rainworm: Conjunctive Query Finite Determinacy Is Undecidable. In *Symp. on Principles of Database Systems (PODS)*.

Erich Grädel. 2001. Why are Modal Logics so Robustly Decidable? In *Current Trends in Theoretical Computer Science*. 393–408.

Erich Grädel, Colin Hirsch, and Martin Otto. 2002. Back and forth between guarded and modal logics. *ACM Trans. Computational Logic* 3, 3 (2002), 418–463.

Erich Grädel, Martin Otto, and Eric Rosen. 1999. Undecidability results on two-variable logics. *Archive for Mathematical Logic* 38, 4-5 (1999), 313–354.

Erich Grädel and Igor Walukiewicz. 1999. Guarded Fixed Point Logic. In *Symp. on Logic In Computer Science (LICS)*.

Eva Hoogland, Maarten Marx, and Martin Otto. 1999. Beth Definability for the Guarded Fragment. In *Logic Programming and Automated Reasoning (LPAR)*.

David Janin and Igor Walukiewicz. 1996. On the Expressive Completeness of the Propositional μ-Calculus w.r.t. Monadic Second-Order Logic. In *Intl. Conf. on Concurrency Theory (CONCUR)*.

Maarten Marx and Maarten de Rijke. 2005. Semantic characterizations of navigational XPath. *SIGMOD Record* 34, 2 (2005), 41–46.

Michael Mortimer. 1975. On languages with two variables. *Zeitschrift für mathematische Logik und Grundlagen der Mathematik* 21, 8 (1975), 135–140.

Alan Nash, Luc Segoufin, and Victor Vianu. 2010. Views and queries: Determinacy and rewriting. *ACM Trans. Database Syst.* 35, 3 (2010), 21:1–21:41.

Martin Otto. 2010. Highly Acyclic Groups, Hypergraph Covers and the Guarded Fragment. In *Symp. on Logic In Computer Science (LICS)*.

Martin Otto. 2013. Expressive completeness through logically tractable models. *Annals of Pure and Applied Logic* 164, 13 (2013), 1418–1453.

Eric Rosen. 1997. Modal logic over finite structures. *Journal of Logic, Language, and Computation* 6, 4 (1997), 427–439.

Balder ten Cate and Luc Segoufin. 2013. Unary negation. *Logical Methods in Computer Science (LMCS)* 9 (2013).

Balder ten Cate and Luc Segoufin. 2015. Craig Interpolation for bounded variable fragments of guarded negation. (2015). Unpublished manuscript.

Johan van Benthem. 1983. *Modal Logic and Classical Logic*. Bibliopolis.

Moshe Y. Vardi. 1996. Why is Modal Logic So Robustly Decidable?. In *Descriptive Complexity and Finite Models*. 149–184.

Moshe Y. Vardi. 1998. Reasoning about The Past with Two-Way Automata. In *Intl. Colloquium on Automata, Languages and Programming (ICALP)*.

Klaus W. Wagner. 1987. More Complicated Questions about Maxima and Minima, and some Closures of NP. *Theoretical Computer Science* 51, 1-2 (1987), 53 – 80.

COMPLEXITY COLUMN

NEIL IMMERMAN, University of Massachusetts Amherst
immerman@cs.umass.edu

We have learned a great deal about algorithms and computational complexity over the last half century. However, the lack of progress answering certain questions has been stunning. In particular, we have huge trouble proving lower bounds, i.e., that we need at least a certain amount of time or space to solve a particular problem. Here is a beautifully written survey by Ryan Williams explaining some new approaches with which he and his coauthors have achieved state-of-the-art complexity lower bounds. The methods – using the potential power of circuits against themselves – should be particularly appealing to logicians.

Some Ways of Thinking Algorithmically About Impossibility[1]

Ryan Williams,
MIT Computer Science and AI Laboratory, Cambridge, MA, USA

Computational complexity lower bounds like $\mathsf{P} \neq \mathsf{NP}$ assert impossibility results for all possible programs of some restricted form. As there are presently enormous gaps in our lower bound knowledge, a central question on the minds of today's complexity theorists is *how will we find better ways to reason about all efficient programs?*

I argue that some progress can be made by (very deliberately) thinking *algorithmically* about lower bounds. Slightly more precisely, to prove a lower bound against some class $\mathcal{C}$ of programs, we can start by treating $\mathcal{C}$ as a set of inputs to another (larger) process, which is intended to perform some basic analysis of programs in $\mathcal{C}$. By carefully studying the algorithmic "meta-analysis" of programs in $\mathcal{C}$, we can learn more about the *limitations* of the programs being analyzed.

This essay is mostly self-contained; scant knowledge is assumed of the reader.

1. INTRODUCTION

We use the term *lower bound* to denote an assertion about the computational intractability of a problem. For example, the assertion "factoring integers of 2048 bits cannot be done with a Boolean circuit of 10^6 gates" is a lower bound which we hope is true (or at least, if the lower bound is false, we hope that parties with sinister motivations have not managed to find such a magical circuit).

The general problem of mathematically proving computational lower bounds is largely a mystery. The stability of modern commerce relies on certain lower bounds being true (most prominently in cryptography and computer security). Yet for practically all of the prominent lower bound problems, we do not know how to begin proving them true—we do not even know *step zero*. (For some major open problems, such as the Permanent versus Determinant problem in arithmetic complexity [Mulmuley 2012], we do have good candidates for step zero, and possibly step one.) Many present lower bound conjectures may well be false. In spite of considerable intuitions that we have about lower bounds, we must admit that our formal understanding of them is awfully weak. This translates to a lack of understanding about algorithms as well.

In this article, I describe two recently developed ways of viewing lower bound problems in a constructive, algorithmic way. Of course the *idea* of viewing lower bounds in this way is not at all new. A good example from mathematical logic is the framework of Ehrenfeucht-Frässé games [Frässé 1950; Ehrenfeucht 1961], with which one can prove that certain core problems cannot be expressed in certain logics, by con-

[1]**This article is an expanded version of an invited article by the author that appeared in the proceedings of the 25th EACSL Annual Conference on Computer Science Logic.** Supported in part by NSF CCF-1212372 and a Sloan Fellowship. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

structing winning strategies for a special two-player game on logical structures. The methods described here are rather different from EF games, and are deeply rooted in the language and methods of worst-case algorithm design.

1.1. Barriers

Why are lower bounds so difficult to prove? There are formal reasons, which are often called "complexity barriers." These are theorems which demonstrate that the usual tools for reasoning about computability and lower bounds—such as universal simulation—are simply too abstract to distinguish modes of computation like P and NP. There are three major classes of barriers known.

Relativization, Algebrization, Natural Proofs. Many ways of reasoning about algorithm complexity remain valid when one adds "oracles" to the computational model: that is, one adds an instruction that can call an *arbitrary* function $\mathcal{O} : \{0,1\}^\star \to \{0,1\}$ in one step, as a black box. When a proof of a theorem is true no matter which $\mathcal{O}$ is added to the instruction set of the relevant computational model(s), we say that the proof "relativizes." Relativizing proofs of statements are generally quite powerful and broadly applicable. However, relativizing proofs are of limited use in computational complexity lower bounds: for instance, $\mathsf{P} = \mathsf{NP}$ when some oracles $\mathcal{O}$ are added to polynomial-time and nondeterministic-polynomial-time algorithms (picking a powerful enough oracle will do), but $\mathsf{P} \neq \mathsf{NP}$ when some other oracles $\mathcal{O}'$ are added (as proved by Baker, Gill, Solovay [Baker et al. 1975]). Therefore, no proof resolving P versus NP can be a relativizing proof. Practically all other major open problems in lower bounds exhibit a similar resistance to arbitrary oracles, and a surprisingly large fraction of theorems in complexity theory do relativize.

The more recently developed "algebrization" barrier [Aaronson and Wigderson 2009] teaches a similar lesson, applied to a broader set of algebraic techniques that was designed to get around relativization. (Instead of looking at oracles, they look at more general algebraic objects, where an oracle O that takes n-bit inputs can be "lifted" to a low-degree polynomial $\tilde{\mathcal{O}} : \mathbb{F}^n \to \mathbb{F}$ over a field $\mathbb{F}$ of order greater than two.) These barriers teach us that, in order to make progress on problems resembling P versus NP, it will be necessary to study the guts of programs at a somewhat low level, and reason more closely about their abilities relative to their simple instruction sets.[2]

The Razborov-Rudich "natural proofs" barrier [Razborov and Rudich 1997] has a more subtle pedagogical point compared to the other two. Informally, they show that strong lower bound proofs *cannot* produce a polynomial-time algorithm for determining whether a given function is hard or easy to compute—otherwise, such an algorithm would (in a formal sense) refute *stronger* lower bounds that we also believe to hold. It turns out that many lower bound proofs from the 1980s and 1990s have such an algorithm embedded in them.

1.2. Intuition and Counter-Intuition

There are also strong intuitive reasons for why lower bounds are hard to prove. The most common one is that it seems extraordinarily difficult to reason effectively about the infinite set of all possible efficient programs, including programs that *we will never see or execute*, and argue that none of them can solve an NP-complete problem. Based on this train of thought, some famous computer scientists such as Donald Knuth have dubbed problems like P versus NP to be "unknowable" [Knuth 2015].

[2]There are definitely "non-relativizing" and even "non-algebrizing" techniques in complexity theory, but they are a minority; see Section 3.4 in Arora and Barak [Arora and Barak 2009] for more discussion.

But how difficult is it, really, to reason about all possible efficient programs? Let us give some counter-intuition, which will build up to the main point of this article. We begin with the observation that, while reasoning about lower bounds appears to be difficult, reasoning about *worst-case algorithms* does not appear to be. Reasoning computationally about an infinite number of finite objects is commonplace in the analysis of worst-case algorithms. There, we have some computable function $f : \Sigma^\star \to \Sigma^\star$ that we would like to compute faster, and one proves that a particular efficient procedure P outputs $f(x)$ on *all possible* finite inputs x. That is, often in algorithm analysis we manage to reason about all possible finite inputs x, even those x that we will never see or encounter in the real world. Our idea is that, if we can find "meta-computational" problems which

(a) treat their inputs as *programs*,
(b) determine interesting properties of the function computed by the input program, and
(c) have interesting/non-trivial algorithms,

then we can hope to import ideas from the design and analysis of algorithms into the theory of complexity lower bounds. (Yes, this is vague, but it is counter-*intuition*, after all.)

Sanity Check: Computability Theory. We must be careful with this counter-intuition. Which computational problems actually satisfy those three conditions? Undergraduate computability theory (namely, Rice's theorem [Rice 1953]) tells us that, if our inputs x encode programs that take arbitrarily long inputs, then it is undecidable to determine non-trivial properties of the function being computed by the program x. Thus, it would seem that any problem that satisfies conditions (a) and (b) above will fail to satisfy condition (c).

One source of this undecidability is the "arbitrary length" of inputs. In particular, if x encodes a program that takes only *finitely* many inputs, say only inputs of length n, then one can produce the entire finite function computed by the input x, and decide non-trivial properties of that function in a computable way. To simplify the discussion (and without significantly losing generality), we might as well think of the input x as encoding a Boolean logic circuit over AND, OR, and NOT gates, taking some n bits of input and outputting some m bits. Now, x simply encodes a directed acyclic graph with additional labels on its nodes, and a procedure P operating on x's can be said to be a "circuit analysis" program, which reasons about the aggregate behavior of finite circuits on their inputs.

However, one of the major lessons of the theory of NP-hardness is that, while reasoning about arbitrary programs may be undecidable, reasoning about arbitrary circuits *is* often decidable but is still highly likely to be intractable. Probably the simplest possible circuit analysis problem is:

Given a Boolean circuit C, does C compute the all-zeroes function?

This problem is already very difficult to solve; it is equivalent to the NP-complete problem CIRCUIT SATISFIABILITY (a.k.a. Circuit SAT) which asks if there is some input on which C outputs 1. From this point of view, the assertion $\mathsf{P} \neq \mathsf{NP}$ tells us that arbitrary programs are intractable to analyze, even over finitely many inputs: we cannot feasibly determine if a given circuit is *trivial* or not. As circuit complexity is inherently tied to P versus NP, the assertion $\mathsf{P} \neq \mathsf{NP}$ appears to have negative consequences for its own provability; this looks depressing. (This particular intuition has been proposed many times before; for instance, the Razborov-Rudich work on "natural proofs" [Razborov and Rudich 1997] may be viewed as one way to formalize it.)

Slightly Faster SAT Algorithms? The hypothesis $\mathsf{P} \neq \mathsf{NP}$ only says that *very* efficient circuit analysis is impossible. More precisely, for circuits C with k bits of input encoded in n bits, $\mathsf{P} \neq \mathsf{NP}$ means there is no $k^{O(1)} \cdot n^{O(1)}$ time algorithm for detecting if C is satisfiable. There is a giant gap between this kind of bound and the $2^k \cdot n^{O(1)}$ time bound obtained by simple exhaustive search over all 2^k possible inputs to the circuit. What if we simply asked for a *non-trivial running time* for detecting the satisfiability of C, something that is merely faster than exhaustive search?

I would like to argue that finding any asymptotic improvement over 2^k time for CIRCUIT SATISFIABILITY is already a very interesting problem. This is not an obvious point to argue. First off, without any further knowledge of its inner workings, a $1.9^k \cdot n^{O(1)}$-time algorithm for CIRCUIT SATISFIABILITY would not be terribly more useful in *practice* than the $2^k \cdot n^{O(1)}$ time of exhaustive search: one would only see a different for small values of k, and the rest of the instances would remain intractable.[3] Work on worst-case algorithms for SAT over many years (such as [Monien and Speckenmeyer 1979; Dantsin 1981; Monien and Speckenmeyer 1985; Kullmann and Luckhardt 1997; Pudlák 1998; Hirsch 2000; Schöning 2002; Paturi et al. 2005; Wahlström 2007; Chen et al. CC14], see the survey [Dantsin and Hirsch 2009]) has been primarily motivated by the intrinsic interest in understanding whether trivial exhaustive search is optimal for solving the Satisfiability problem.

The Non-Black-Box-ness of Circuit SAT Algorithms. There is also a deeper reason to pursue minor improvements in SAT algorithms. Any algorithm for CIRCUIT SAT running in (say) time $1.9^k \cdot n^{O(1)}$ must necessarily provide a kind of "non-relativizing" analysis of *every* given circuit C, an analysis which relies on the structure and encoding of C. If you were required to design a CIRCUIT SAT algorithm which could only access C as a black-box *oracle*, obtaining the output $C(y)$ from inputs y and no other information about C, then your algorithm would necessarily require at least 2^k steps in the worst case. The reason is simple: if you are completely blind to the insides of the circuit C, then even a small ($O(k)$ size) circuit could hide a satisfying k-bit input from you.

In more detail, note there is a trivial circuit Z which always outputs 0, and for every k-bit input y, there is an $O(k)$-size circuit Z_y which outputs 1 if and only if the input equals y. Given the code of your black-box SAT algorithm A, one can note the sequence of k-bit inputs $y_1, y_2, \ldots$ that A calls the oracle on, assuming that the oracle circuit keeps outputting 0. At the end of the execution of A, if it has only seen 0-outputs, then A must conclude that the circuit is not satisfiable (otherwise, it would give the wrong answer on the all-zeroes circuit). Furthermore, if A did not query the oracle on some k-bit input y, then the circuit Z_y will be satisfiable and yet A will have concluded it is unsatisfiable. It follows that, for A to be correct on all circuits Z and Z_y, it needs to call the oracle circuit on all 2^k possible inputs.

Therefore, a $1.9^k \cdot n^{O(1)}$ time algorithm for CIRCUIT SAT must necessarily use the fully-given representation of the circuit in some critical ways, to work faster than exhaustive search. Even an algorithm running in $O(2^k/k)$ time on circuits of size $O(k)$ would be interesting, for the same reason.[4]

---

[3]This attitude is not shared in cryptography, where any improvement in exhaustive search over all keys may be considered a "break" in the cryptosystem.

[4]Perhaps you do not believe that CIRCUIT SAT can be solved any faster than $2^k \cdot n^{O(1)}$ steps. This belief turns out to be inessential for the main intuition and the formal theorems that follow. For example, you may instead believe that we can non-deterministically approximate the fraction of satisfying assignments to a k-input circuit of size n in $1.9^k \cdot n^{O(1)}$ time; this is also something that oracle access to a circuit cannot accomplish. Furthermore, if you do not believe even this, then your lack of faith in algorithms requires you to have

A Possible Road to Circuit Complexity Lower Bounds. The ability to analyze a given circuit more efficiently than analyzing a black box suggests a further implication: a CIRCUIT SAT algorithm running faster than exhaustive search could potentially be used to prove a circuit complexity *limitation*. At the very least, if the CIRCUIT SAT problem can be solved faster than exhaustive search on a given collection of circuits $\mathcal{C}$ (some of which encode the all-zero function, and some which do not), then the collection $\mathcal{C}$ *fails* to obfuscate the all-zeroes function from some algorithm running in less than 2^k steps. That is, the assumed CIRCUIT SAT algorithm can "efficiently" distinguish all circuits encoding the all-zeroes function from those circuits which do not; these circuit cannot hide satisfying inputs as well as oracles can. This points to a potential deficiency in $\mathcal{C}$ that the CIRCUIT SAT algorithm takes advantage of. Surprisingly, this intuitive viewpoint can be made formal.

Outline. In the remainder of this article, we first describe some known connections between circuit satisfiability algorithms and circuit complexity lower bounds (Section 2). Then, we turn to a more recent example of how algorithms and lower bounds are tied to each other, in a way that we believe should be of interest to the union of logicians and computer scientists (Section 3). In particular, we reconsider the basic problem of testing circuit functionality via input-output examples, define the *test complexity* as a way of measuring the difficulty of testing, and describe how circuit complexity lower bounds are *equivalent* to test complexity upper bounds. We conclude the article with some hopeful thoughts.

2. CIRCUIT SAT VERSUS CIRCUIT COMPLEXITY

Let us briefly review some relevant notions from the theory of circuit complexity; for more, see the textbook of Arora and Barak ([Arora and Barak 2009], Chapter 6).

Circuit Complexity. A Boolean circuit with n inputs and one output is a directed acyclic graph with n sources and one sink, and labels of AND/OR/NOT on all other nodes. Each circuit computes some finite function $f : \{0,1\}^n \to \{0,1\}$. To compute infinite languages, of the form $L : \{0,1\}^\star \to \{0,1\}$, the computational model is extended to an infinite family of circuits $\mathcal{F} = \{C_n\}_{n=1}^{\infty}$, where C_n has n inputs and one output. For such a family $\mathcal{F}$, we say that $\mathcal{F}$ *computes* L if for all $x \in \{0,1\}^\star$ we have $C_{|x|}(x) = L(x)$. For a function $s : \mathbb{N} \to \mathbb{N}$, a family $\mathcal{F}$ has *size* $s(n)$ if for all n, the number of nodes (i.e., gates) in C_n is at most $s(n)$. A language L *has polynomial-size circuits* if there is a polynomial $p(n)$ and a family $\mathcal{C}$ of size $p(n)$ that computes L. The class of all languages having polynomial-size circuits is denoted by P/poly. One should think of P/poly as the class of computations for which the minimum "sizes" of computations do not grow considerably with the input length to those computations.

The class P/poly is poorly understood. It could be enormously powerful, or it could be fairly weak. It is easy to see that every language of the form $\{1^n \mid n \in S\}$ (for some subset $S \subseteq \mathbb{N}$) is in P/poly; however, a simple counting argument shows there are undecidable languages of this form. Therefore P/poly contains some undecidable languages. In that sense, P/poly is powerful, but this comes from the fact that the computational model defining P/poly can have infinite-length descriptions. (This observation also shows that traditional thought in computability theory is probably not going to be very helpful in understanding the power of P/poly.) However, P/poly also looks obviously limited, in the sense that for all input lengths n, only polynomial-in-n resources need to be spent in order to decide all 2^n inputs of that length. A counting

strong beliefs in the power of Boolean circuits—they would be powerful enough to solve nondeterministic exponential-time problems. See the references [Impagliazzo et al. 2002; Williams 2013a].

argument shows that for every n, some function $f : \{0,1\}^n \to \{0,1\}$ requires circuits of size exponential-in-n; in fact, *most* functions do.

A prominent question in complexity theory is:

> *How does* P/poly *relate to the Turing-based classes of classical complexity theory, like* P, NP, PSPACE, *etc.?*

It is pretty easy to see that $\mathsf{P} \subset \mathsf{P/poly}$: every "finite segment" of a polynomial-time algorithm can be simulated with a polynomial-size circuit. It is conjectured that $\mathsf{NP} \not\subset \mathsf{P/poly}$ (which would in turn imply $\mathsf{P} \neq \mathsf{NP}$). But it is an open problem to prove that $\mathsf{NEXP} \not\subset \mathsf{P/poly}$! That is, every language in the *exponential-time version of* NP could in fact have polynomial-size circuits. It is amazing that a problem like this is still open. Fifty years ago, Hartmanis and Stearns [Hartmanis and Stearns 1965] showed that some $O(n^3)$-time computations are more powerful than all $O(n)$-time ones; how is it possible that we can't distinguish exponential time Turing machines from polynomial size circuits? This open problem demonstrates how truly difficult it is to prove lower bounds on circuit complexity; perhaps the infinite circuit model is powerful!

2.1. Enter Circuit SAT

Let's return to thinking about the role of CIRCUIT SATISFIABILITY. Earlier, we were arguing that a faster algorithm for Circuit SAT would point to some deficiency in the power of circuits: unlike a black-box oracle, circuits would be unable to successfully "hide" satisfying inputs from algorithms that run in $o(2^k)$ steps. We would like to say:

The *existence* of a "faster" algorithm A solving CIRCUIT SAT, for *all* circuits C from a class of circuits $\mathcal{C}$

$$\Longrightarrow$$

The *existence* of an "interesting" function $f : \{0,1\}^\star \to \{0,1\}$, that is not computable by *all* circuit families from that class $\mathcal{C}$

Written in the above way, the logical quantifiers match up well, and the idea of using an algorithm to prove a circuit complexity lower bound looks less counter-intuitive. Indeed, we can say a formal statement as described in the above box. Here is one version.

THEOREM 2.1 ([WILLIAMS 2013A; 2014B]). *Suppose for all polynomials p,* Circuit Satisfiability *of circuits with n inputs and $p(n)$ size is decidable in $O(2^n/n^{10})$ time. Then* $\mathsf{NEXP} \not\subset \mathsf{P/poly}$. *That is, there are (explicit) functions computable in nondeterministic exponential time that do not have polynomial-size circuits.*

(The polynomial n^{10} is almost certainly not optimal, and it depends on the precise machine model, but it suffices.) Notice the required improvement over exhaustive search: it would normally take $2^n \cdot p(n)$ time to solve SAT on circuits of $p(n)$ size. In order to solve satisfiability fast enough, we need to be able to "divide by an arbitrary polynomial" in the running time. This is a much weaker requirement than bounds like 1.9^n time, which had been the primary focus of researchers.

Here we will briefly describe the proof of Theorem 2.1. For more technical details, see the original paper [Williams 2013a] and the surveys [Williams 2011; Santhanam 2012; Oliveira 2013].

Proof Outline of Theorem 2.1. The known proofs proceed by contradiction. We assume:

(A) There is a CIRCUIT SAT algorithm A running in $2^n/n^{10}$ time, and
(B) Every function $f \in \mathsf{NEXP}$ has polynomial-size circuits. (Note that it is equivalent to assume that a single function, complete for NEXP under polynomial-time reductions, is computable with polynomial-size circuits.)

These two assumptions are inherently algorithmic in nature: item (A) asserts that CIRCUIT SATISFIABILITY can be solved faster, and item (B) asserts that a huge class of decidable problems are computable with polynomial-size circuit families.

The main idea is to utilize these two algorithmic assumptions to solve NEXP-complete problems in a way which is provably *impossible*. Namely, assumptions (A) and (B) together are shown to imply that *every* function computable in nondeterministic time $O(2^n)$ is computable in nondeterministic time $O(2^n/n^2)$, which contradicts the time hierarchy theorem for nondeterminism [Žák 1983].

Compressible Witnesses for NEXP.. Our proof uses a highly non-trivial consequence of item (B). Recall that NEXP consists of those problems L for which there is a verifier algorithm V which runs in time exponential in $|x|$ so that for all inputs x, $x \in L$ if and only if there is a witness y of length exponential in $|x|$ such that $V(x, y)$ accepts. (The definition of NP would replace "exponential" with "polynomial.")

Work of Impagliazzo, Kabanets, and Wigderson [Impagliazzo et al. 2002] shows an important consequence of item (B): if NEXP is in P/poly, then all exponential-time verifiers have *highly compressible* witnesses. More precisely, they prove that item (B) implies:

(C) For all $L \in \mathsf{NEXP}$ and all *exponential-time verifiers V for L*, for every $x \in L$, there is a poly$(|x|)$-size circuit C_x with poly$(|x|)$ inputs with truth table y such that $V(x, y)$ accepts.[5]

In other words, every string x in L has a witness that has a very succinct representation, of only poly(n) size.

The Impossible Algorithm.. Let L be an arbitrary language that is computable in nondeterministic 2^n time. We want to construct another nondeterministic algorithm that computes L in $O(2^n/n^2)$ time, which implies the desired contradiction (as mentioned above). Here is a high-level outline of the nondeterministic algorithm:

Algorithm Outline: On input x,

(1) Nondeterministically *guess* poly(n)-size circuit C_x which is intended to encode a witness for x (Note that if $x \in L$ then such a C_x exists, by item (C).)
(2) Deterministically *verify* that C_x is a correct guess. Here we wish to use the CIRCUIT SAT algorithm asserted by item (A).

Clearly, Step 1 takes poly(n) time to guess the circuit C_x. We need to find conditions for which Step 2 will run in $O(2^n/n^2)$ time, assuming the CIRCUIT SAT algorithm of item (A).

The deterministic verification of Step 2 is a subtle process. The CIRCUIT SAT algorithm of item (A) can check in $O(2^n/n^{10})$ time whether a given n-input circuit always outputs 0, over all possible 2^n input assignments. A key observation is that it would suffice to build a larger circuit D_x with $m \leq n + 8 \log n$ inputs and poly(n) size (in $O(2^n/n^2)$ time) that is based on C_x, such that C_x encodes a witness for x if and only

[5] We use the notation poly(n) to denote an arbitrary fixed polynomial factor of n.

if D_x is not satisfiable. Then, running the CIRCUIT SAT algorithm of item (A) on D_x takes time

$$O\left(2^m/m^{10}\right) \leq O\left(2^{n+8\log(n)}/(n+8\log n)^{10}\right) \leq O\left(2^n/n^2\right),$$

and would be unsatisfiable if and only if C_x encodes a witness for x, as desired. Hence, if we could build such a D_x, then we could implement step 2 of the above algorithm outline in time $O(2^n/n^2)$.

How to Use the Circuit SAT Algorithm.. So how can we build a circuit D_x which is unsatisfiable if and only if C_x encodes a witness for x? We turn to special structural properties of NEXP computations. In particular, every language L computable in nondeterministic $O(2^n)$ time can be (very) efficiently reduced to the following NEXP-complete problem:

> SUCCINCT 3SAT: *Given a circuit E of n inputs, does its truth table of 2^n bits encode a satisfiable 3CNF formula?*

In particular, as we evaluate E on various inputs, the outputs encode *clauses* of an exponentially-large 3CNF formula. Sharp technical results on the NP-hardness of 3SAT [Fortnow et al. 2005; Williams 2013a] show how to reduce any nondeterministic $O(2^n)$-time L to a poly(n)-size instance of SUCCINCT 3SAT with at most $n + 4\log n$ inputs. That is, we have a polynomial-time algorithm A that, given an instance x, outputs a circuit E_x which is a yes-instance of SUCCINCT 3SAT if and only if $x \in L$.

From here, the desired circuit D_x can be constructed in the following way. Note that the witness circuit C_x guessed for an instance of SUCCINCT 3SAT will encode a variable assignment. On an input y of length $n + 4\log n$, our circuit D_x:

— Prints the "yth clause" of the 3CNF formula encoded by E_x.
— Uses three copies of the guessed circuit C_x to verify if at least one of the three literals in the yth clause is satisfied by the assignment encoded by C_x.
— Outputs 0 if and only if the yth clause is indeed satisfied.

Then, such a D_x is unsatisfiable if and only if C_x encodes a satisfying assignment to the 3CNF formula encoded by E_x. This completes the proof outline of Theorem 2.1. □

While the above proof yields the desired outcome, it is obviously an indirect method, and feels lacking. It is an interesting open problem to find simpler and/or more informative proofs of this algorithms-to-lower-bounds connection.

Applying the Connections. The framework behind Theorem 2.1 has been generalized so that circuit satisfiability algorithms for various circuit classes $\mathcal{C}$ imply lower bounds for computing functions in NEXP with circuits from $\mathcal{C}$. So far, through the design of new circuit satisfiability algorithms, this framework has led to several unconditional circuit lower bound results:

— NEXP does not have so-called ACC^0 circuits of polynomial size [Williams 2014b],
— $\mathsf{NE}/1 \cap \mathsf{coNE}/1$ (a potentially weaker class) does not have ACC^0 circuits of polynomial size [Williams 2013b].
— NEXP does not have ACC^0 circuits of polynomial size, augmented with a layer of neurons (linear threshold gates) that connect directly to the inputs [Williams 2014a].
— E^{NP} does not have sub-quadratic size circuits composed of arbitrary symmetric and linear threshold gates [Tamaki 2016].

— E^{NP} does not have ACC^0 circuits of $2^{n^{o(1)}}$ size, augmented with a layer of $n^{2-\varepsilon}$ neurons (linear threshold gates) that connect to another layer of $2^{n^{o(1)}}$ neurons, and this second layer connects directly to the inputs [Alman et al. 2016].

All results except for the second were obtained by designing explicit circuit satisfiability algorithms for the relevant circuit classes; the second was obtained by sharpening the complexity-theoretic arguments themselves.

One might not believe that even slightly faster circuit satisfiability algorithms are possible. We stress that it is possible that one might prove that $\mathsf{NEXP} \not\subset \mathsf{P/poly}$ without providing a new CIRCUIT SAT algorithm. There are at least two ways in which this could be done:

1. Instead of solving the NP-hard CIRCUIT SAT problem, it is possible to show [Williams 2013a; Santhanam and Williams 2014] that one only needs to deterministically solve the following CIRCUIT APPROXIMATION PROBABILITY PROBLEM (CAPP):

> *Given a circuit C that is promised to have either at least 2^{n-1} satisfying assignments, or zero satisfying assignments, determine which is the case.*

With randomness, it is trivial to distinguish between the two cases with high probability, even in a black-box way: just pick uniform random inputs and check if any of them make C output 1. By using succinct probabilistically checkable proofs in place of SUCCINCT 3SAT [Ben-Sasson et al. 2005; Ben-Sasson and Viola 2014], a deterministic solution to CAPP running in $2^n/n^{10}$ time on circuits of polynomial size would also imply $\mathsf{NEXP} \not\subset \mathsf{P/poly}$.

2. It could be that the *assumption* $\mathsf{NEXP} \subset \mathsf{P/poly}$ could be used to imply the existence of satisfiability algorithms sufficient for proving $\mathsf{NEXP} \not\subset \mathsf{P/poly}$. (It is in fact known that $\mathsf{NEXP} \subset \mathsf{P/poly}$ implies faster algorithms for solving some NP problems, but CIRCUIT SAT is not known to be among them!)

3. CIRCUIT COMPLEXITY AND TESTING CIRCUITS WITH DATA

We now turn to a problem related to program verification, and describe a connection to circuit complexity. In practice, programs are often verified by the quick-and-dirty method of trial and error: the program is executed on a suite of carefully chosen inputs, and one checks that the outputs of the program are what is expected. For a given function to compute, it is natural to ask when trial and error can be efficient: when does can one use a small number of input-output examples, and determine correctness of the program with certainty?

If there are no constraints on what the program can be, then there is not much to say about this problem: without any further information, the program is a black box, and one would simply have to try all possible inputs to know whether the program does what it should. But in testing, we're never given a program as a black box; we know *something* about it, such as its total size. Could such side information be useful for the testing problem? Formal mathematical work on this kind of problem from years ago (such as [Howden 1976; DeMillo and Lipton 1978; Budd and Angluin 1982]) focused on restrictions to the functions to be computed and how to efficiently generate test data for them, rather than including more side information about the programs themselves to make the testing problem less black-box.

Recently with Brynmor Chapman [Chapman and Williams 2015], we have proposed a general circuit-analysis problem that we call *data design*. Let $f : \{0,1\}^\star \to \{0,1\}$

be a function to test for, and a class $\mathcal{C}$ of size-s circuits that can implement "slices" of f (restricted to fixed-length inputs). The task of *data design for* f is to select a small suite of input-output *test data* that can be used to determine whether a given n-input circuit C from $\mathcal{C}$ computes f restricted to n-bit inputs.

More formally, we say that the *test complexity* of f (as a function of the circuit size s) is the minimum number of input-output examples such that, for any circuit C of size s, one can determine with certainty whether C computes f (on all n-bit inputs, where n is the number of inputs to C), by evaluating the circuit C on the examples. Note that every function f depending on all of its inputs has test complexity $O(s2^s)$: the test suite may contain all possible input-output pairs for f on all input lengths $n = 1, \ldots, s$. When can test suites be made smaller?

While the data design problem certainly has practical motivation, we are mainly interested in the problem due to its intriguing inversion of the roles of program and input. The circuit C computing f is the *input* to the data design problem, and the *program* for testing the circuit is the collection of input-output examples needed to determine if C computes a slice of f. We have uncovered a surprising correspondence between upper bounds on data design and lower bounds on circuit complexity. In general:

> Designing small suites of data for testing whether $\mathcal{C}$-circuits compute f
> is *equivalent* to
> Proving $\mathcal{C}$-circuit lower bounds on computing f.

For example:

THEOREM 3.1 ([CHAPMAN AND WILLIAMS 2015]). *A function f is in $\mathsf{P}/poly$ if and only if for some $\varepsilon > 0$, the test complexity of testing circuits for f is greater than 2^{s^ε} for almost every s.*

So if we wanted to prove that (for example) that $\mathsf{NEXP} \not\subset \mathsf{P}/\mathrm{poly}$, it would be necessary and sufficient to design test suites of sub-exponential test complexity for a function in NEXP. Intuitively, such a correspondence is possible because the circuit design problem and the data design problems work with similar types of unknowns. The circuit C designed must compute f on all n-bit inputs, and the set of data designed must test the functionality of f for all s-size circuits C. Let us outline the ideas of the proof.

Proof Outline of Theorem 3.1. There are two parts to the equivalence.

Circuit Complexity Upper Bounds Imply Test Complexity Lower Bounds.. Let $k \geq 1$. If the function f has a circuit of size n^k on n-bit inputs, then a test suite for size-s circuits must include at least $2^{\Omega(s^{1/k})}$ input-output pairs. That is, an upper bound on the circuit complexity of f implies a *lower bound* on the test complexity of f.

To prove this, we observe a couple of simple facts. First, if f has a circuit of size n^k on n-bit inputs, then there is some circuit C of size s that computes f on inputs of length about $s^{1/k}$. Second, for every input y of length $s^{1/k}$, there is a circuit C_y of about $s + O(s^{1/k})$ size which equals C on all inputs except for y. From these two facts, one can show that the test suite for size-s circuits will need to include all possible input-output pairs for f on $(\delta \cdot s^{1/k})$-bit inputs (for some $\delta > 0$), in order to distinguish the good size-s circuit C from all of the other (bad) C_y circuits. That is, the test suite needs at least $2^{\delta s^{1/k}}$ input-output pairs.

Circuit Complexity Lower Bounds Imply Test Complexity Upper Bounds.. For the other side of the equivalence, we show that if f does not have a circuit of size $O(n^{k+1})$ on n-bit inputs, then there is a set of only poly(n^k) input-output pairs $(x, f(x))$ on n-bit inputs that is sufficient for testing all circuits of size up to n^k. (This statement is something of a "dual" to a result in learning theory [Bshouty et al. 1996] on learning small circuits with an NP oracle.) Then, a lower bound on the circuit complexity of f for circuits of size n^{k+1} ensures that the test complexity for f is not too large for testing circuits of size n^k.

The above result applies theorems on "sparse" strategies for zero-sum games. (Recall that for zero-sum games, a *strategy* is just a probability distribution on the set of possible actions for a player.) Such theorems [Althöfer 1994; Lipton and Young 1994] say that for every zero-sum game where the "row player" has m possible actions and the "column player" has n possible actions, there is a strategy for the row player with only $O(\log n)$-size support and for the column player with only $O(\log m)$-size support, that closely approximates the optimal strategy of the game.

To apply such results here, the key idea is to consider a zero-sum game with a Circuit Player (ranging over all circuits of size up to n^k) and an Input Player (ranging over all inputs of length up to n). If f does not have a circuit of size $O(n^{k+1})$ on n-bit inputs, then for every tuple $(C_1, \ldots, C_{O(n)})$ of n^k-size circuits that could form a sparse strategy for the Circuit Player, there is an input $x^\star$ such that $f(x^\star) \neq MAJORITY(C_1(x^\star), \ldots, C_{O(n)}(x^\star))$. That is, every sparse strategy of the Circuit Player badly fails to compute f on at least one input $x^\star$: for every sparse strategy, a random choice of a circuit from the strategy fails to compute f on $x^\star$ with probability very close to $1/2$.

As every sparse strategy for the Circuit Player badly fails to compute f, one can show that there must be a good sparse strategy for the Input Player: there is a poly(n^k)-size set S of input-output pairs $(x, f(x))$ such that for every circuit C of size up to n^k, there is an $(x, f(x)) \in S$ such that $C(x) \neq f(x)$. This set S is precisely the test suite that we wanted to obtain.

Let f_n be the restriction of f to inputs of length n. Putting the two above items together, we (informally) have

$$\begin{aligned} f \in \mathsf{P/poly} &\iff (\exists k)(\forall n)[f_n \text{ has } n^k\text{-size circuits}] \\ &\iff (\exists k)(\forall s)[\text{the test complexity of } f \text{ is } 2^{\Omega(s^{1/k})}] \end{aligned}$$

This concludes the proof outline. □

Designing reliable exhaustive circuit tests and circuit complexity lower bounds are therefore deeply related tasks, when phrased in the above language. Not only would small test suites detect errors efficiently, they would also be useful for formal verification. Assuming the circuit being tested is in the appropriate class $\mathcal{C}$, passing a test suite would be a proof of correctness on all inputs. In turn, proving that a small test suite (relative to the circuit size) works is equivalent to proving a limitation on what can be computed in $\mathcal{C}$. This "constructive" algorithmic viewpoint on lower bounds is still in its early stages of development, and it remains to be seen how effectively one can prove lower bounds with it.

4. CONCLUSION

Knowledge from all areas of theoretical computer science could contribute significantly to the general projects outlined here. Computer scientists will have to develop new methods of argument in order to make a serious dent in the major lower bound problems, and it is worth trying every sort of reasonable argument we can think of (at least

once). Perhaps the logic side of computer science will provide some of these new proof methods.

REFERENCES

Scott Aaronson and Avi Wigderson. 2009. Algebrization: A New Barrier in Complexity Theory. *ACM TOCT* 1 (2009). Issue 1.

Josh Alman, Timothy M. Chan, and R. Ryan Williams. 2016. Polynomial Representations of Threshold Functions and Algorithmic Applications. In *FOCS*. 467–476.

Ingo Althöfer. 1994. On sparse approximations to randomized strategies and convex combinations. *Linear Algebra Appl.* 199 (1994), 339–355.

Sanjeev Arora and Boaz Barak. 2009. *Computational Complexity - A Modern Approach*. Cambridge University Press.

Theodore Baker, John Gill, and Robert Solovay. 1975. Relativizations of the P =? NP Question. *SIAM J. Comput.* 4, 4 (1975), 431–442.

Eli Ben-Sasson, Oded Goldreich, Prahladh Harsha, Madhu Sudan, and Salil P. Vadhan. 2005. Short PCPs Verifiable in Polylogarithmic Time. In *IEEE Conference on Computational Complexity (CCC*. 120–134.

Eli Ben-Sasson and Emanuele Viola. 2014. Short PCPs with projection queries. In *ICALP*. 163–173.

Nader Bshouty, Richard Cleve, Ricard Gavalda, Sampath Kannan, and Christino Tamon. 1996. Oracles and Queries that are Sufficient for Exact Learning. *J. Comput. System Sci.* 52, 2 (1996), 268–286.

Timothy A. Budd and Dana Angluin. 1982. Two Notions of Correctness and Their Relation to Testing. *Acta Informatica* 18 (1982), 31–45.

Brynmor Chapman and Ryan Williams. 2015. The Circuit-Input Game, Natural Proofs, and Testing Circuits With Data. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS*. 263–270.

Ruiwen Chen, Valentine Kabanets, Antonina Kolokolova, Ronen Shaltiel, and David Zuckerman. 2015. See CCC'14. Mining Circuit Lower Bound Proofs for Meta-Algorithms. *Computational Complexity* 24, 2 (2015. See CCC'14), 333–392.

Evgeny Dantsin. 1981. Two propositional proof systems based on the splitting method. *Zapiski Nauchnykh Seminarov LOMI* 105 (1981), 24–44.

Evgeny Dantsin and Edward A. Hirsch. 2009. Worst-Case Upper Bounds. In *Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications*. Vol. 185. IOS Press, 403–424.

Richard A. DeMillo and Richard J. Lipton. 1978. A probabilistic remark on algebraic program testing. *Inform. Process. Lett.* 7, 4 (1978), 192–195.

Andrzej Ehrenfeucht. 1961. An application of games to the completeness problem for formalized theories. *Fundamenta Mathematicae* 49 (1961), 129–141.

Lance Fortnow, Richard J. Lipton, Dieter van Melkebeek, and Anastasios Viglas. 2005. Time-space lower bounds for satisfiability. *JACM* 52, 6 (2005), 835–865.

Roland Frässé. 1950. Sur une nouvelle classification des systmes de relations. *Comptes Rendus* 230 (1950), 1022–1024.

Juris Hartmanis and Richard Stearns. 1965. On the Computational Complexity of Algorithms. *Trans. Amer. Math. Soc. (AMS)* 117 (1965), 285–306.

Edward A. Hirsch. 2000. New Worst-Case Upper Bounds for SAT. *J. Autom. Reasoning* 24, 4 (2000), 397–420. DOI:http://dx.doi.org/10.1023/A:1006340920104

W. E. Howden. 1976. Reliability of the path analysis testing strategy. *IEEE Transactions on Software Engineering SE-2* 3 (1976), 208–214.

Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. 2002. In search of an easy witness: Exponential time vs. probabilistic polynomial time. *J. Comput. Syst. Sci.* 65, 4 (2002), 672–694.

Donald E. Knuth. 2015. Personal communication. (2015).

Oliver Kullmann and Horst Luckhardt. 1997. Deciding propositional tautologies: Algorithms and their complexity. Technical report, Fachbereich Mathematik, Johann Wolfgang Goethe Universität. (1997).

Richard J. Lipton and Neal E. Young. 1994. Simple strategies for large zero-sum games with applications to complexity theory. 734–740.

Burkhard Monien and Ewald Speckenmeyer. 1979. 3-satisfiability is testable in $O(1.62^r)$ steps. Technical Report Bericht Nr. 3/1979, Reihe Theoretische Informatik, Universität-Gesamthochschule-Paderborn. (1979).

Burkhard Monien and Ewald Speckenmeyer. 1985. Solving Satisfiability in Less Than 2^n Steps. *Discrete Applied Mathematics* 10, 3 (1985), 287–295.

Ketan Mulmuley. 2012. The GCT program toward the *P* vs. *NP* problem. *Commun. ACM* 55, 6 (2012), 98–107. DOI:http://dx.doi.org/10.1145/2184319.2184341

Igor Oliveira. 2013. *Algorithms versus Circuit Lower Bounds*. Technical Report TR13-117. ECCC.

Ramamohan Paturi, Pavel Pudlák, Michael E. Saks, and Francis Zane. 2005. An improved exponential-time algorithm for k-SAT. *JACM* 52, 3 (2005), 337–364.

Pavel Pudlák. 1998. Satisfiability - Algorithms and Logic. In *Mathematical Foundations of Computer Science 1998, 23rd International Symposium, (MFCS'98)*. 129–141. DOI:http://dx.doi.org/10.1007/BFb0055762

Alexander Razborov and Steven Rudich. 1997. Natural Proofs. *J. Comput. Syst. Sci.* 55, 1 (1997), 24–35.

H. G. Rice. 1953. Classes of Recursively Enumerable Sets and Their Decision Problems. *Trans. Amer. Math. Soc.* 74 (1953), 358–366.

Rahul Santhanam. 2012. Ironic Complicity: Satisfiability Algorithms and Circuit Lower Bounds. *Bulletin of the EATCS* 106 (2012), 31–52.

Rahul Santhanam and Ryan Williams. 2014. On Uniformity and Circuit Lower Bounds. *Computational Complexity* 23, 2 (2014), 177–205.

Uwe Schöning. 2002. A Probabilistic Algorithm for k-SAT Based on Limited Local Search and Restart. *Algorithmica* 32, 4 (2002), 615–623.

Suguru Tamaki. 2016. A Satisfiability Algorithm for Depth Two Circuits with a Sub-Quadratic Number of Symmetric and Threshold Gates. *Electronic Colloquium on Computational Complexity (ECCC)* 23 (2016), 100.

Magnus Wahlström. 2007. *Algorithms, Measures, and Upper Bounds for Satisfiability and Related Problems*. Ph.D. Dissertation. Linköping University.

Ryan Williams. 2011. Guest column: a casual tour around a circuit complexity bound. *ACM SIGACT News* 42, 3 (2011), 54–76.

Ryan Williams. 2013a. Improving Exhaustive Search Implies Superpolynomial Lower Bounds. *SIAM J. Comput.* 42, 3 (2013), 1218–1244.

Ryan Williams. 2013b. Natural Proofs Versus Derandomization. In *STOC*. 21–30.

Ryan Williams. 2014a. New algorithms and lower bounds for circuits with linear threshold gates. In *STOC*. 194–202.

Ryan Williams. 2014b. Nonuniform ACC Circuit Lower Bounds. *JACM* 61, 1 (2014), 2.

Stanislav Žák. 1983. A Turing machine time hierarchy. *Theoretical Computer Science* 26, 3 (1983), 327–333.

SEMANTICS COLUMN

MICHAEL MISLOVE, Tulane University
mwm@math.tulane.edu

This quarter's Semantics column features an article by Jean Krivine, a CNRS researcher who is a member of the Proofs, Programs and Systems group that forms a part of the Institute for Fundamental Research in Informatics (IRIF) at Université Paris Diderot. Jean is leader in the application of computer science to Systems Biology, a broad area whose research activities range from biologists in wet labs, to computational scientists devising simulations based on mathematical models, to computer scientists developing models based on computer science models and their underlying principles. Jean's column highlights the tension that arises when wet lab researchers, who are accustomed to time-tested research methodologies, confront new approaches driven by novel applications of computer science and its principles to the problems they face. The column also describes the issues computer scientists face if they want to have a significant impact in a related discipline. Last but surely not least, Jean's description of the Kappa language and its application to protein interactions provides an impressive example of how concurrency theory and techniques developed within computer science can usefully be applied in a related discipline.

This seems a good opportunity to comment on developments at the interface between computer science and the myriad of related areas where computer science has been having a disruptive effect on how research is conducted. During its development in the latter half of the 20th century, as with any evolving new science, computer scientists focused on adapting approaches and techniques from other disciplines to solve problems in their own area. But with the turn of the century, examples began to emerge of computer scientists attacking problems in related disciplines using the techniques and underlying principles of computer science. Two remarkable examples arose within the semantics community: one was the development of categorical models of quantum mechanics, an approach grounded in category theory and informed by experience with computational models (cf. [6; 7; 1]). The second is the topic of this quarter's column: the application of computer science, and especially of concurrency theory to Systems Biology. As part of his presentation, Jean outlines contributions by colleagues such as Luca Cardelli and his co-researchers at Microsoft Cambridge, Vincent Danos and colleagues in Edinburgh and Paris, and of course the contributions he and his colleagues in Paris have made. The common ingredient shared by these approaches is that they all are grounded in programming language semantics, and those such as the Kappa language that Jean has developed, as well as examples such as Biocham [4] and Cardelli's strand graph rewriting approach [5] utilize rewriting techniques to model biological processes such as protein interactions. Such approaches often have a stochastic component, to interpret the rates at which interactions occur between the many possible

participating proteins. Sometimes, as in the work of Cardelli, differential equations are used to model the evolution of a system over time.

Jean has led the development of the *Kappa* language, a prototypical example using concurrency and rewriting techniques to model cascades of protein interactions. But before telling us about Kappa and how it is applied, Jean first sets the stage by describing how wet lab biologists perceive the new technologies computer science has to offer. Here Jean draws on his experience in Walter Fontana's Systems Biology Lab at Harvard to offer insights into how systems biology research is conducted. In pointing out that new, unproven technologies must win over wet lab researchers if they are to be adopted, Jean makes a clear case that these novel computer science-based approaches need to demonstrate their effectiveness before the traditional biology research community will adopt them *en masse*. Jean also proposes a rôle for which he believes Kappa and similar approaches are best suited to address the needs of Systems Biology.

From the time I first became aware of efforts to apply concurrency theory to Systems Biology, I've thought that this is a particularly apt example for demonstrating the maturation of computer science as a scientific discipline. Indeed, one of the models of the concurrent lambda calculus that emerged in the 1990s was Berry and Boudol's *Chemical abstract machine* (CHAM) [3], itself inspired by a language for parallel multiset computation [2] whose authors cited the following chemical analogy. The CHAM model was based on viewing computational processes as molecules in a "chemical soup", where interactions – synchronizations – between individual processes could be seen as similar to how molecules in such a soup bond to form new chemicals. This model was particularly appealing at a time when the process calculus community was just coming to grips with notions of mobility. A leader here was Robin Milner, with his work on mobile ambients. This has a direct connection with Jean's work, because Jean worked with Robin on the development of the bigraph model.

The first time I heard Jean speak about Kappa and its rule-based approach to modeling protein interactions, I was reminded of Berry and Boudol's CHAM model. It strikes me as particularly *apropos* that this column shows how the earlier adaptation by the concurrency community of an approach from chemistry can be turned on its head by an approach using principles from programming semantics and concurrency to model biological interactions. I am particularly pleased to welcome Jean's column, and to learn not only how Kappa can be applied in Systems Biology, but also to learn the hurdles applying such an approach faces, and to hear the advice of a leader on what Kappa and similar approaches need to do to have an impact in this area.

REFERENCES

1. Abramsky, Samson and Bob Coecke, A categorical semantics of quantum protocols, Proceedings of the 19th IEEE conference on Logic in Computer Science (LiCS'04). IEEE Computer Science Press (2004).
2. Banâtre, J.-P., Coulant, A., and D. Le Metayer, A parallel machine for multiset transformation and its programming style, *Future Generation Computer Systems* **4** (1988), pp. 133-144.
3. Berry, Gèrard and Gèrard Boudol, The chemical abstract machine, *Theoretical Computer Science* **96** (1992), pp. 217–248.
4. The Biocham web page: https://lifeware.inria.fr/biocham/
5. Cardelli, L., Strand algebras for DNA computing, Natural Computing **10** (2011), pp. 407–428.
6. Kelly, G.M. and M.L. Laplaza, Coherence for compact closed categories, Journal of Pure and Applied Algebra 19, 193213 (1980).
7. Selinger, P., Dagger compact closed categories and completely positive maps, Proceedings of the 3rd International Workshop on Quantum Programming Languages, Chicago, June 30–July 1 (2005).

Systems Biology

Jean Krivine, IRIF, CNRS & Université Paris Diderot

1. INTRODUCTION

1.1. Big data

The unequal race between technological and theoretical innovation in Computer Science has a mirror image in Systems Biology where the balance is even more biased toward the experimental side. The aim of Systems Biology is to produce a specification of natural systems, when artificial systems are usually already equipped with one[1]. As a consequence, Systems Biology is almost exclusively an experimental science where most of the innovation effort is geared toward the conception of devices able to extract more facts from biological systems. It has resulted in a modern trend of Systems Biology where high-throughput experiments are now producing big data and a massive inflation of publication volume (Figure 1). The lack of comprehensive integration of

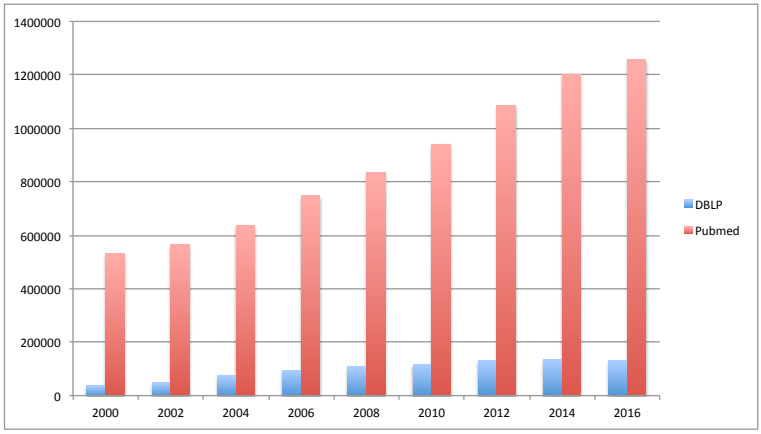


Fig. 1. Conference and workshop papers referenced by DBLP since 2000 (blue) vs. journal papers referenced by PubMed (red), the main database of papers published in biology and medicine.

biological knowledge is hidden by the success stories of synthetic design using bio material: the initial aim of understanding how the basic components of the cell conspire in a complex and changing environment is modestly replaced by the science of hijacking cellular and genetic components in order to implement intelligently designed tasks. In

[1]Some properties of artificial systems are not specified. For instance the growth of the internet can be studied as an experimental science [Huberman and Adamic 1999].

turn these bio-devices enable biologists to set up more experiments and accumulate more data. This life cycle between technological innovation and data accumulation is encouraged by high impact journals in biology, which tend to give more credit to technological breakthroughs than (disputable) progress in understanding the ways of the cell.

1.2. Reductionism and Systems biology

> *"Big mechanisms are large, explanatory models of complicated systems in which interactions have important causal effects. The collection of big data is increasingly automated, but the creation of big mechanisms remains a human endeavor made increasingly difficult by the fragmentation and distribution of knowledge. To the extent that the construction of big mechanisms can be automated, it could change how science is done".* Paul Cohen, DARPA 2014.

The above citation is excerpt from a presentation of DARPA's "Big Mechanism" project that was launched in 2014 aimed at the production of mechanistic models of the cell, grounded on big data [You 2015]. This $45M project was based on a reductionist approach to better understand cell regulation. The plan was to collate protein-protein interaction data from biological papers using automated deep reading techniques, and integrate the extracted nuggets of biological information into large executable models. What are these "facts", the collection of which is big data, and that attracts so much attention from the molecular biologists? For a large part, they are protein-protein and protein-DNA/RNA interactions. For the reader of this column, proteins can be viewed as agents with internal states, able to bind to each other and induce state change in so doing (see Figure 2).

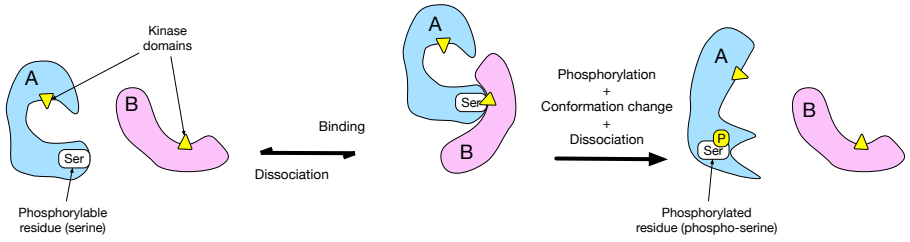


Fig. 2. Two proteins A and B able to assemble into a protein complex (a heterodimer in biological terms). Yellow triangles denote *kinase domains* (a domain is a sub-part of a protein) able to catalyze *phosphorylation*, the attachment of a phosphoryl group to an amino acid residue such as serine (Ser). A protein bearing a kinase domain is called a *(protein) kinase*, so both A and B are kinases. We say that B is *active*, meaning that its kinase domain is accessible to its environment; A is *inactive* in both the left and middle state. The phosphorylation of A by B "activates" A: its kinase domain becomes accessible after a conformation change of the protein enabled by phosphorylation. Another consequence of the conformation change is that B is no longer able to bind to A, although B is still active and may phosphorylate other proteins.

Importantly the stability or mere feasibility of a protein complex is conditioned by the structural conformation of its constituents, which in turn may depend on their internal states. For instance the protein *Cellular tumor antigen p53* (p53 for short) has 23 amino-acid residues that can undergo chemical modifications, called *post-transcriptional modifications* (PTMs). This entails that p53 has about 2^{23} possible internal states (some of its residues have actually several possible modifications). In the example of Figure 2, protein A has two internal states defined by its serine residue being phosphorylated or not. Protein A also has two structural conformations, inactive

(closed form) or active (open form), which correlate with the internal states *unphosphorylated* and *phosphorylated* respectively[2].

Biologists try to uncover correlations between conformation of a protein and its PTMs in order to use them as proxys to express their "activity". On `Uniprot.org`, one of the main online databases of proteins, at the p53 entry one can read for instance:

"Phosphorylation at Ser-9 by HIPK4 increases repression activity on BIRC5 promoter."

Translated here, this sentence indicates that the modification of the ninth residue of p53 (which is a Serine) improves the stability of p53 on DNA (thanks to a favorable conformation change), in the vicinity of BIRC5 (a gene) promoter, in turn interfering with the transcription of this gene. The sentence further indicates that this phosphorylation is catalyzed by *Homeodomain-interacting protein kinase 4* (HIPK4 for short).

By studying, enumerating and quantifying protein-protein and protein-DNA interactions, biologists are trying to understand *signaling pathways* of the cell. One might expect here a formal definition of these pathways but sadly none really exists. The least we can say is that a pathway is a collection of protein-protein interactions (binding, modifications, conformation change,...) equipped with annotations describing positive or negative influence between interactions. Amidst those interactions, some initial events, called *signals*, are distinguished. They occur at the surface of the cell and correspond to the binding of a chemical element (a protein or sometimes a smaller molecule) to *receptor* proteins that are encased and diffusing on the cell membrane. Biologists say that signaling pathways implement *signal transductions*, which describe how the initial signals are propagated down to the nucleus (by means of protein-protein interactions), where they eventually trigger genetic "responses" (a collection of genes are being transcribed into RNA). Typical cellular responses to signals are cell growth and division (for instance after reception of the *Epidermial Growth Factor* (EGF) signal) [Oda et al. 2005], cell death (for instance as a consequence of the reception of signals that belong the the *Tumor Necrosis Factor* family) [Houston and O'Connell 2004], chemotaxis (bacteria cell moving toward nutrients) [Van Haastert and Devreotes 2004] etc.

Mutations of genes, the product of which are involved in growth or death signal transductions, are strongly correlated with cancer development. For instance *Mitogen Activated Protein Kinases* (MAPK) cascades, which are core signaling components of the cell, are currently the target of most therapeutic efforts [Arslan et al. 2006].

1.3. Knowledge overflow.

Now, let us imagine the scientific life of a systems biologist working on MAPK cascades such as EGFR signaling (cell growth), searching for potential new targets for chemotherapy. In theory, she should be aware of the 373 proteins involved in the pathway[3] and the corresponding protein-protein interactions (binding, PTMs, translocation,...). To illustrate the complexity of her task, we give in Figure 3 a representation of these interactions as referenced on KEGG, a biological database of pathways[4].

[2]One should really use correlation here, since thermodynamically speaking it is not impossible that unphosphorylated A spontaneously shifts from the closed to the open form.
[3]Source: `reactome.org`, online database of biological pathways.
[4]http://www.genome.jp/kegg/pathway.html

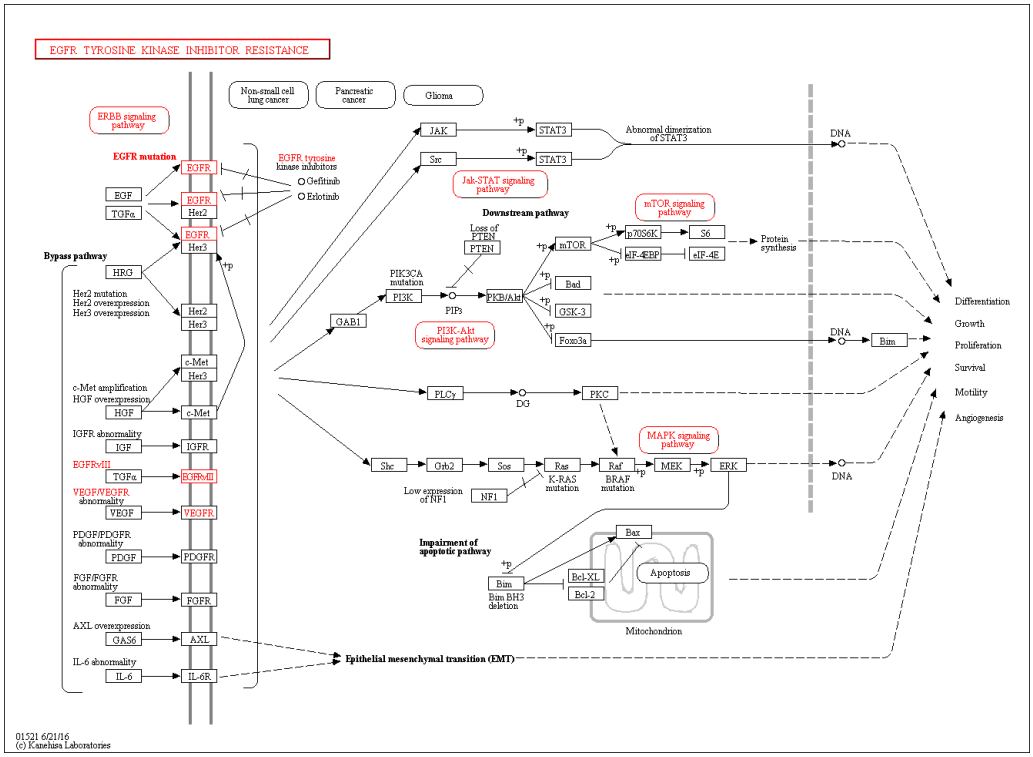


Fig. 3. A representation of a MAPK signaling pathway (here EGFR signaling), as pictured on the KEGG database of pathways. Rectangular nodes denote proteins small, circles denote other molecules such as metabolites. Regular arrows denote activation and T-shaped arrows denote inhibition. +p labelled arrows denote phosphorylation. Red nodes point to other pathway maps (source KEGG).

Other databases of signaling pathways, containing maps with different conventions and different emphasis on the components of the pathways are accessible to biologists [Chowdhury and Sarkar 2015]. Several points should be noted about these maps:

— They are essentially *qualitative*: although they have either a Petri Net or event structure flavor, these maps are not really equipped with an operational semantics, although some offer possible compilations into differential equations. They are mainly designed to depict biological knowledge about the pathway and point the biologists to relevant literature, each reaction being labelled with a reference paper.
— They are *semi-formal*: as we mentioned in the introduction of this column, protein-protein interactions are highly combinatorial. It is hence not possible to have a extensional representation of these interactions, less even so one that can be visualized. As a consequence informal choices are made by the curator of the pathway when pruning which complexes (and which internal states) should be explicitly included in the database.
— They have a *maintenance problem*: because these maps should reflect state of the art knowledge about pathways, they suffer from an obsolescence problem, the speed of which depends on the research activity pertaining to the pathway of interest. We give,

in Figure 4, an illustration of this problem showing the increasing rate of publication related to MAPK cascades (papers containing "MAPK" *in the title*)[5].

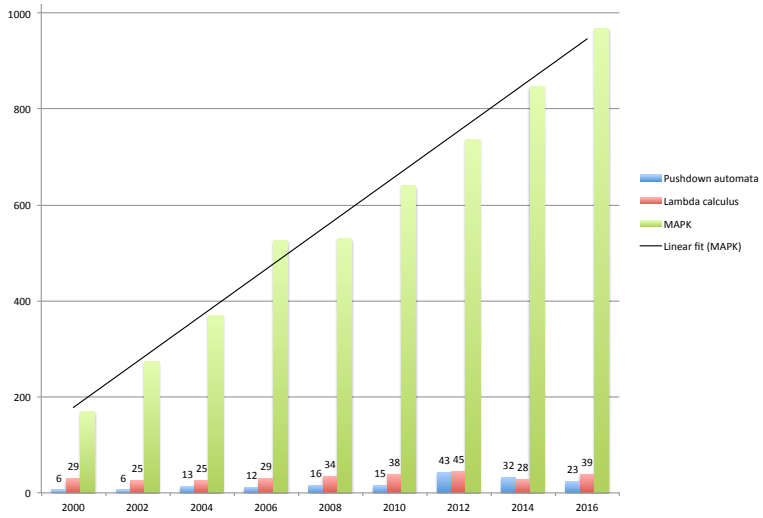


Fig. 4. Yearly scientific production of papers mentioning MAPK cascade in its title. In 2008 over 500 papers pertaining to MAPK were being published (source `pubmed.org`). For comparison, the same year 34 λ-calculus papers and 16 Pushdown automata papers are referenced (source `dblp.org`). The acceleration of publication (derivative of the best linear fit) pertaining to MAPK is about a 100 papers per year.

Even the most specialized biologist in MAPK signaling cannot reasonably be aware of all publications and new observations pertaining to her topic of interest. Biologists are thus filtering information out of the big data of scientific literature. Pragmatically, this selection is implemented by three consecutive filters described in Figure 5. These filters can be understood as a necessary evil when the activity of modeling is, for the most part, implicit and where models are required to hold in human brains. But, to rephrase DARPA, if biological big data has underlying "big mechanisms", there is little hope that any mental construction can ever comprehend these "large explanatory models". Moreover, because of the combinatorics of protein-protein interactions, no static map, database, or ontology list, can suffice to complement human brains in the design of these models. These simple observations have taken a large swath of molecular biologists away from any (formal or semi-formal) modeling activity. From our experience most experimental biologists give little credit to existing models, which they know are partial, outdated and have little predictive power. Yet, while rejecting models, experimental biologists have to rely on implicit mental constructs in order to decide which experiment to set up. Thus *every* biologist has a model, either implicitly or explicitly. We believe that the lack of success story of formal models in Systems biology is not due to the impossibility of having accurate and useful models, but rather to the fact that one is confusing *models* of interactions with *explanations* of biological functions. Our approach is to view models as programs and biological functions as computations. If models are to be treated as programs we need a language to assemble them.

[5]It is noteworthy that the last update of the EGFR signaling pathway on the `reactome.org` database was performed in 2008.

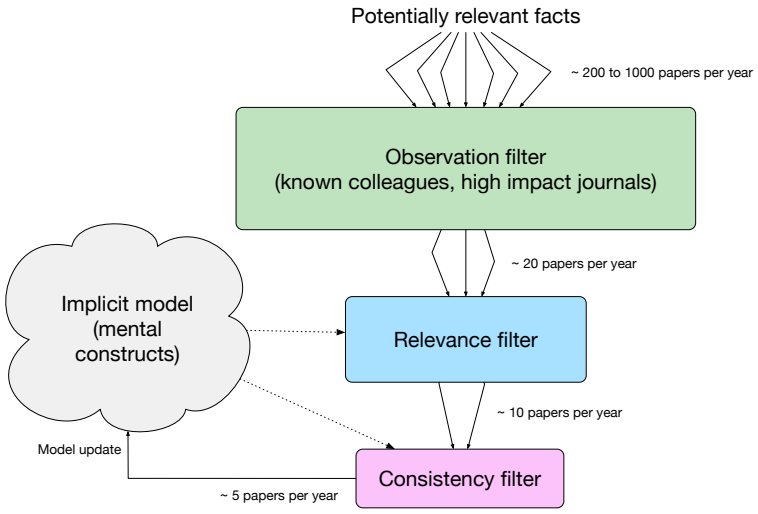


Fig. 5. A model of the selection procedure of a biologist. The first (observation) filter is passive: a biologist only considers papers that she is aware of. This is realized by focusing her readings on high impact journals, or by monitoring well-known colleagues activity. The next two filters are active and rely on an implicit model of the pathway that the biologist has in mind (possibly seconded by maps such as the one of Figure 3). The relevance filter consists in neglecting papers that do not contain facts that are of immediate impact on the implicit model. The consistency filter is the (sometimes unconscious) process of rejecting papers that do not confirm the implicit model. In a subtle manner, papers that are rejected this way still have an impact on the implicit model, by weakening the beliefs the biologist has on some contradictory facts.

2. A LANGUAGE FOR SYSTEMS BIOLOGY

2.1. Fixing the abstraction barrier

Designing a domain specific language (DSL) is not a simple problem: in the case of protein-protein interactions, it amounts to deciding which biological event should be considered a basic construct of the language, and which ones should be *abstracted away*. How does one express that protein A is in an "open" conformation? Can one translate the fact that "smoking causes cancer"? Said another way, we need to pick where to place the abstraction barrier. As depicted in Figure 6, this choice is not innocent. Pushing the abstraction in the intensional direction allows one to rely on a small

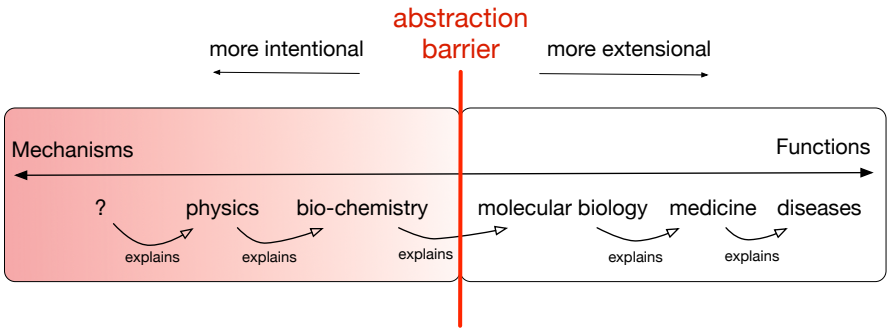


Fig. 6. Choosing where to place the abstraction barrier.

number of transformation laws. In the limit, everything boils down to energy minimization, which should *explain* anything that happens at higher levels of abstraction. However having a few laws to describe the evolution of a system comes at a price: observing a macroscopic event requires too many computation steps. The time scales of a computational biologist, modeling protein folding, is in nano seconds, whereas clocks of signaling pathways tick in hours. Going in the other direction offers efficient computations: one define state changes that are macroscopic and on time frames that are compatible with potential drug interactions. However the prize here is the explosion of the "laws" of the world: in the extensional limit, very little can be explained since almost everything is here as if by decree.

Placing an abstraction barrier implies that a fact, which could have been explained by mechanisms that have been abstracted away, becomes a *rule* of the model [Hlavacek et al. 2006]. We call here an *assertion*, a biological observation which can be made anywhere on the mechanistic/functional scale of Figure 6. An assertion pertaining to mechanisms that occur before the abstraction barrier will naturally be interpreted as a rule of the model ("A forms a covalent bond with B"), while assertions on the functional side need to be verified by *executions* of the model ("HIV virus causes immunosuppression"). Once the language is fixed, the activity of modeling Systems biology consists in collating biological facts, which can be directly encoded as rules, and functional assertions, which the model should eventually be able to explain. We will come back to this important distinction later, with an example. In the meantime we will introduce Kappa, a formalism that corresponds to a particular choice of abstraction that is intended to leave as much space as possible for explanations while remaining efficiently executable.

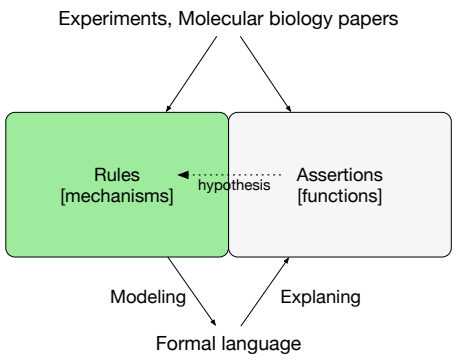


Fig. 7. Model assembly: collecting mechanistic rules and biological assertions. Rules can be used to implement the model, which is in turn used to validate the assertions. If the model fails to validate all of them, hypothesis can be formulated in order to turn assertions into possible rules.

2.2. Kappa in a nutshell

The need for an executable modeling framework for systems biology has had some publicity in the systems biology literature [Regev and Shapiro 2002; Hlavacek et al. 2006; Fisher and Henzinger 2007], but clearly the most active community is to be found within theoretical computer science. Several approaches, stemming from a theoretical background, have been proposed to provide executable representations of biological systems at various levels of abstraction. Boolean networks [Thomas 1973; Kauffman

et al. 2003; Paulevé and Richard 2012; Comet et al. 2013; Ruet 2015], multi-set rewriting [Chabrier and Fages 2003; Pedersen and Plotkin 2008; Păun and Romero-Campero 2008; John et al. 2011] and process algebra [Danos and Krivine 2003; Phillips and Cardelli 2007; Ciocchetta and Hillston 2009] have been considered for static analysis, model checking, simulation or simply representation.

When fixing the abstraction barrier just after bio-chemistry (see Figure 6), it is tempting to model protein complexes as graphs where nodes would represent proteins and where edges would denote physical contact between them. Several graph rewriting formalisms have been considered in that direction [Danos and Laneve 2003; Faeder et al. 2004; Krivine et al. 2008; Andrei and Kirchner 2007; Damgaard et al. 2012]. Kappa is a graph rewriting language [Danos et al. 2007] that is tailored to represent molecular biology at the level of protein interactions. It is equipped with a stochastic rewriting strategy that is implemented in an efficient simulator [Danos et al. 2007; Boutillier et al. 2017a], which makes Kappa a good target for compiling knowledge representation formalisms [Lopez et al. 2013; Basso-Blandin et al. 2016]. We dedicate the remainder of this section to a brief introduction to the Kappa formalism.

Kappa graphs. Assume a set $K = \{A, B, C, \dots\} \uplus \{\varepsilon\}$ of molecule *sorts*, with a distinguished element ε called *none*; and a set of *agents* $\mathcal{A} = \{a, b, c, \dots\}$ that are sorted by the map $\kappa : \mathcal{A} \to K$. Kappa nodes are agents equipped with *sites* through which connections are made. Formally, we consider a *signature* map $\Sigma : K \to \mathbb{N}$, with $\Sigma(\varepsilon) =_{\mathsf{def}} 1$, that defines how many sites an agent has, and the set of Kappa *nodes* $\mathcal{N}_\Sigma \subseteq \mathcal{A} \times \mathbb{N}$ satisfies $(a, i) \in \mathcal{N}_\Sigma$ if and only if $i \leq \Sigma(a)$. An *edge* is a 2-element set of the form $e = \{(a, i), (b, j)\}$. Two edges e and e' are *coherent*[6] if and only if $e \cap e' \neq \emptyset$ implies $e = e'$. This allows one to capture the fact that binding sites are resources: once an agent a is bound on its site i, it can no longer participate in another bond through i.

A Kappa *graph* is a pair $G = (N_G, E_G)$ where $N_G \subseteq \mathcal{N}_\Sigma$ is a set of nodes and $E_G \subseteq \mathcal{P}_2(N_G)$ is a finite coherent set of edges. We write:

$$G = \{\{(a, 1), (b, 2)\}, \{(b, 1), (c, 1)\}\}_{a:A, b:B, c:\varepsilon}$$

for the graph G that contains the agent a of sort A, bound to agent b of sort B on their sites 1 and 2 respectively, the site 1 of b being free (*i.e.*, bound to an agent of the special sort *none*).

For biological modeling we actually refine sorts into $K = K_{\mathsf{mol}} \uplus K_{\mathsf{state}} \uplus \{\varepsilon\}$ where K_{mol} and K_{state} are used respectively to give names to molecules and PTMs. Agents of the sort K_{state} are also constrained to have only one site, *i.e.*, $A \in K_{\mathsf{state}}$ implies $\Sigma(A) =_{\mathsf{def}} 1$. We give in Figure 8 the graphical notation to represent Kappa graphs.

A Kappa graph *homomorphism* $f : G \to H$ is a map on the agents of G that preserves edges, *i.e.*,:

$$\{(a, i), (b, j)\} \in E_G \implies \{(f(a), i), (f(b), j)\} \in f(E_G)$$

Kappa rules. As discussed in the previous section, Kappa rules are the fundamental components of a model. They describe the laws of the biological system under investigation. Formally a rule r is a pair of maps $r =_{\mathsf{def}} (f : D \to L, g : D \to R)$ where f and g are injective homomorphisms and D, L, R are Kappa graphs called, respectively, the *domain of definition*, the *left hand side* and the *right hand side* of r. For simplicity we do not deal here with node deletions that induce side effects (see Ref. [Danos et al. 2012] for a formal treatment of side effects using single pushout rewriting). We give in Figure 9, diagram **A**, an example of rule application using double pushout rewriting [Raoult 1984].

[6]This definition is taken from Ref [Boutillier et al. 2017a] where graphs are viewed as cliques in a coherent space.

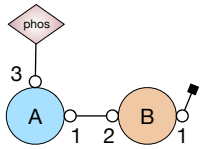


Fig. 8. $G = \{\{(a,3),(c,1)\},\{(a,1),(b,2)\},\{(b,1),(d,1)\}\}_{a:A,b:B,c:\mathsf{phos},d:\varepsilon}$ represented graphically. Small circles denote sites, large circles denote gluings of nodes sharing the same agent of sort K_{mol}, while diamonds denote agents sorted by K_{state}. Black square nodes denote agents of the *none* sort. Sites that are bound to them are called *free*. Sites of agents, the signature of which indicates it has only one site, are omitted (here the sort phos and ε). This example reads as "protein A, which is phosphorylated on its site 3, is bound to protein B (on their sites 1 and 2 respectively). The site 1 of protein B is free".

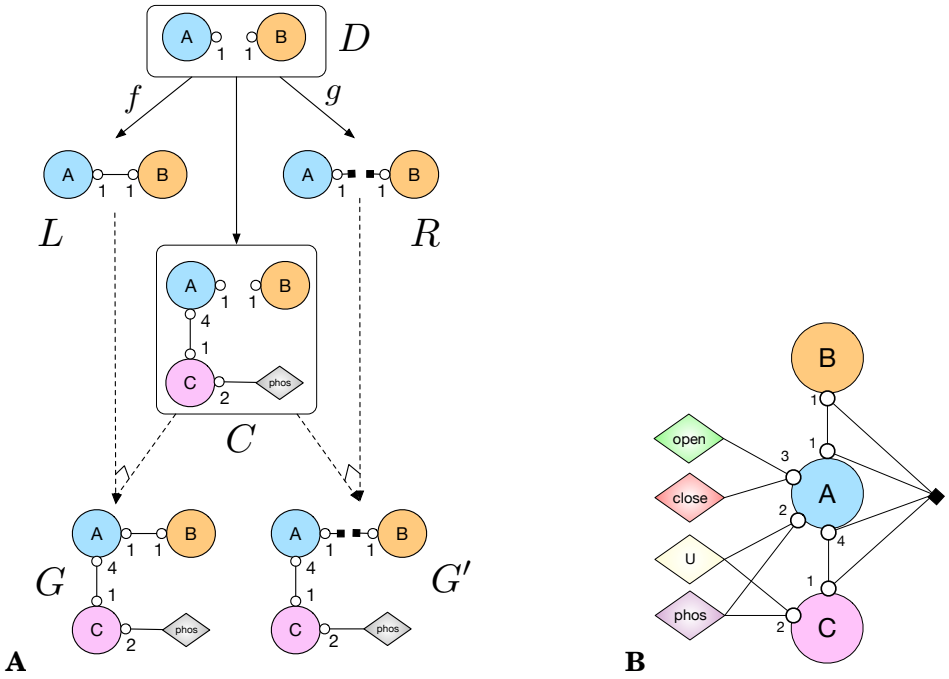


Fig. 9. **A**: An example of a Kappa rule $(f : D \to L, g : D \to R)$. The domain of definition D indicates that only nodes $(a,1)$ and $(b,1)$ are preserved, and that both edges of R are created. The effect of the rule is "transported" into the transition $G \to G'$, *via* a double pushout construction using a context of application C. **B**: A contact map that allows one to type interactions. For instance the site 1 of an agent of sort A may only be bound to the site 1 of an agent of sort B, or free (*i.e.*, bound to an agent of the sort *none*). Both sites 2 of agents of sort A and B can be phosphorylated or unphosphorylated (bound to a phos or u agent respectively).

2.3. Kappa models and simulation

A model in Kappa is a collection agent declarations, specifying the Σ_K map and a set of rules (such as depicted in Figure 9). It is possible to type the rules that are used in the model according to a *contact map* [Danos et al. 2013], an example of which is given in Figure 9, diagram **B**. The contact map of a model can be either statically inferred from the rules, or imposed as a constraint on rule declarations. Models can

be executed using KaSim [Boutillier et al. 2017b], the stochastic simulator of Kappa graph rewriting[7].

In a nutshell, the stochastic rewriting procedure follows the Gillespie algorithm [Gillespie 1976] adapted to rule-based modeling [Danos et al. 2007]. Consider a state G, the Kappa graph that is to be rewritten, and a finite set of Kappa rules $\mathcal{R}$. The algorithm describes (i) the probability that $r \in \mathcal{R}$ is used to rewrite G, and (ii) the probability that one should wait less than a certain δt (in arbitrary time units) in order to see a rewrite taking place. More precisely, the algorithm proceeds as follows:

(1) For all $r \in \mathcal{R}$, compute $[r;G]$, the number of matches (injective homomorphisms) of the left hand side of r in G and define the *activity* at G as $\lambda_G =_{\text{def}} \sum_{r\in\mathcal{R}}([r;G]*k_{r,G})$, where $k_{r,G} \in \mathbb{R}^+$ is the kinetic rate of rule r at state G.
(2) Select rule r with probability $p_{r,G} =_{\text{def}} \frac{[r;G]*k_{r,G}}{\lambda_G}$ and rewrite the graph G by selecting a match of r with probability $\frac{1}{[r;G]}$ (uniform probability).
(3) Draw *time advance* δt, a random variable whose probability density function is $\lambda_G e^{-\lambda_G t}$ (the exponential distribution with parameter λ_G).

Note that after each rewrite $G \to G'$ has taken place, one needs to recompute $[r;G']$ for all rules of the model. To do so, one needs to compute the update vector $\vec{u} \in \mathbb{Z}^k$ where k is the number of rules in $\mathcal{R}$ and u_i, the ith coordinate of $\vec{u}$, is the number of matches of rule i that have appeared or disappeared as a consequence of the rewrite. The update vector can be computed incrementally, using an update data structure, called *extension base*, which is computed statically from the rule set [Boutillier et al. 2017a].

3. MODELS AS PROGRAMS, BIOLOGICAL FUNCTIONS AS COMPUTATIONS

3.1. Rule collation

We wish to illustrate now the activity of a hypothetical modeler, whose goal is to translate biological facts "into" the Kappa language. If we were this modeler, we would be reading biological literature, collating important assertions, in order to translate them into Kappa. For instance:

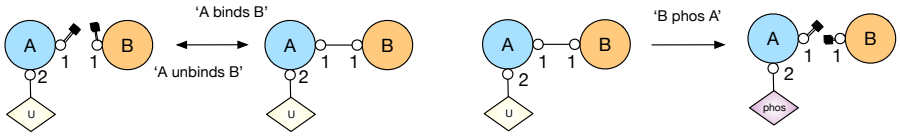
"Kinase B is a highly efficient enzyme that phosphorylates A" (1)

"Kinase A phosphorylates C" (2)

"A has a closed and an open form" (3)

"Kinase B activates A" (4)

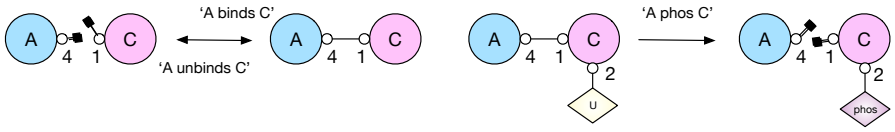
These examples of biological snippets of information are typical of what can be found in a molecular biology paper. We can represent assertion (1) using the following rules (we omit the domain of definition):



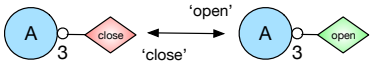
We are making an important assumption here: we implement B's "efficiency" by requiring A to be unphosphorylated in order for B to bind to it (the scheme is similar

[7]The simulator is accessible via the browser at www.kappalanguage.org

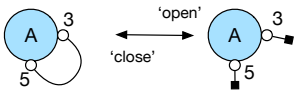
to the sequence of interactions depicted in Figure 2). Since we don't have the same information for assertion (2) we simply use the rules:



Assertion (3) mentions two structural configurations for protein A that we represent with labels:



Note that it would be possible to represent the open and closed conformations this way:



which would be closer to the structural biology of A. However this self-binding of A has no consequence on the rest of the interactions (it does not hide other binding sites), so we stick to the first encoding which is equally expressive: if an interaction should depend on the open conformation of A it suffices to test whether site 3 is bound to an open agent (or in the alternative, to test whether sites 3 and 5 are free).

The assertion (4) is further down the functional axis of Figure 6, compared to the previous ones. It can be rephrased as "The more Kinases B are there, the faster A phosphorylates its substrates". As we argued, showing that a biological fact can be *explained* by the rules of the model is the task of semantics. The simplest form of verification of these assertions is to mimic what a biologist would do in such case: set up an experiment. We can use a simulation to test the speed of A's activity with little B to start with, and check the consequence of adding more B to the activity of A.

To do so, we enter the model into KaSim, the Kappa simulator. We refer the reader to Ref. [Boutillier et al. 2017b] for the actual syntax of the KaSim and we simply reproduce the code below, that essentially contains the rules described above, plus some initial conditions and observables: the ratio of closed A and phosphorylated A, over the total number of A and the ratio of phosphorylated C over the total number of C.

```
##Agent Declarations
%agent: A(x1!x1.B,x2~u~phos,x3~close~open,x4!x1.C)
%agent: B(x1!x1.A)
%agent: C(x1!x4.A,x2~u~phos)

##Rules
'A binds B' A(x1,x2~u),B(x1) <-> A(x1!1,x2~u),B(x1!1) @ 'kon','koff'
'B phos A' A(x1!1,x2~u),B(x1!1) -> A(x1,x2~phos),B(x1) @ 'kphos'

'A binds C' A(x4),C(x1) <-> A(x4!1),C(x1!1) @ 'kon','koff'
'A phos C' A(x4!1),C(x1!1,x2~u) -> A(x4),C(x1,x2~phos) @ 'kphos'

'open A' A(x3~close) <-> A(x3~open) @ 'kopen','kclose'
```

```
##Variables
%var: 'kon' 0.01 # Association rate
%var: 'koff' 1 # Dissociation rate
%var: 'kphos' 1 # Phosphorylation rate
%var: 'kopen' 1 # Rate for opening A
%var: 'kclose' 1 # Rate for closing A

##Initial conditions
%init: 1000 A() # by default, unphosphorylated and closed
%init: 10 B()
%init: 10000 C() # by default, unphosphorylated

##Observables
%obs: 'Aclose' |A(x3~close)|/|A()| #closed A over total number of A's
%obs: 'phos A' |A(x2~phos)|/|A()|  #pho'ylated A's over total number of A's
%obs: 'phos C' |C(x2~phos)|/|C()|  #pho'ylated C's over total number of C's
```

The result of this "in silico" experiment is given in Figure 10: as expected, the amount of B in the system is of no observable consequence on the activity of A, which does not support assertion (4).

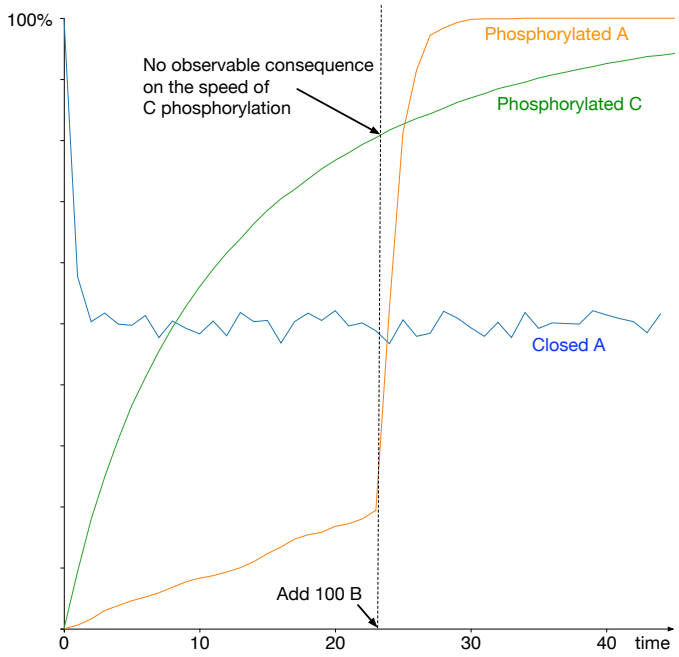
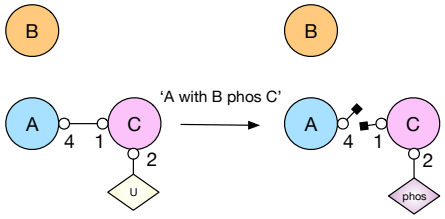


Fig. 10. Experiment to test whether the amount of B has an impact on the activity of A. We initialized the system with 10 Bs, 1,000 As (in a closed configuration, unphosphorylated) and 10,000 Cs (unphosphorylated).

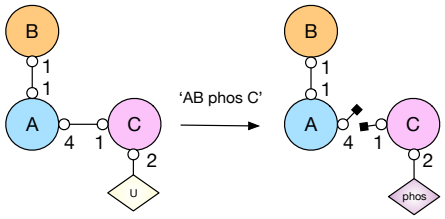
3.2. Hypothesizing rules

In the same way that *checking* a proof of a theorem is easier than *finding* a proof, the problem of realizing an assertion that is not satisfied by a set of rules is a complex

task. In biological modeling however, the difficult part is not to find *some* rules that will realize the assertion in *some* molecular context: one needs to find *plausible* rules that realize the assertion in a *plausible* context. To illustrate this issue with our example, let us assume we try to realize assertion (4) by requiring that kinase B should be present, in order for C to be phosphorylated by A. This can be done by replacing the 'A phos C' rule by:

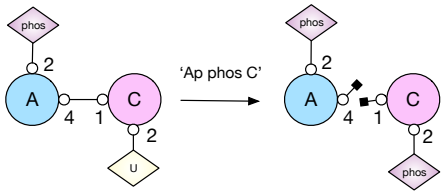


If we assume that there is literally no B in the cell, and then at some point we add a few Bs, we obtain a clear activation (Fig. 11, plot **A**). However, if we run the experiment under more plausible conditions, namely that a few Bs are already present in the system, then adding more Bs does not induce any observable activation of Kinase A (Fig. 11, plot **B**). This is due to the fact that the rule 'A with B phos C' implements the influence of B over A's activity *at a distance*. This non local influence, besides not being biologically sound, would imply that a single B could activate all the As that are present in the cell almost at once. In order to correct for this, one may remark that A and B can form a complex, *via* the rule 'A binds B'. So we can consider the alternative refinement:



but again this rule is only realizing assertion (4) in a weak manner (Fig. 11, plot **C**): indeed, we hypothesized that a conformation change of A, following its phosphorylation, prevents B from binding to A. So the rule 'A binds B' has, at first, a positive effect of C phosphorylation, but also has eventually a negative effect by enabling more A to be phosphorylated. So *via* this simple rule we, in fact, realized two seemingly contradictory assertions: "B activates A" and "B inhibits A".

In order to correlate the presence of B to the activity of A we could also chose to require A to be in its phosphorylated form in order to, in turn, phosphorylate C:



as expected, this rule makes visible the positive impact of B's concentration on A's activity (Fig. 11, plot **D**).

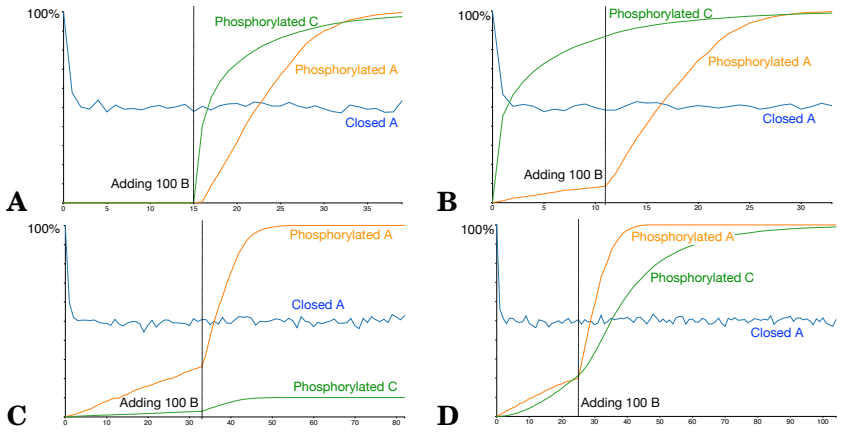


Fig. 11. Testing different variations of the initial model in order to verify "Kinase B activates A", by injecting more B after a few seconds. **A** and **B**: variant of the model using the rule 'A with B phos C' starting with no B (experiment **A**) or 10 B (experiment **B**). **C**: variant of the model using the rule 'AB phos C'. **D**: variant of the model using the rule 'Ap phos C'.

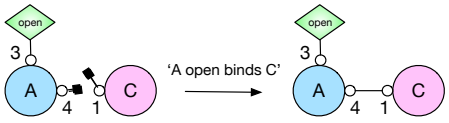
3.3. Model hypothesis and compositionality

Synthesizing proofs requires proof search strategies: it is unlikely one may find interesting theorems by doing a random walk in the space of valid proofs. What is a reasonable strategy to infer new rules in the context of biological modeling? In the previous section we implicitly followed a parsimonious strategy: we do not really hypothesize new mechanisms (a new possible binding, a new PTM, a new protein), we only used a principle of *rule refinement*. With this strategy the only allowed refactoring of a model is the substitution of a rule by a collection of more precise ones, each of them making the application condition of a mechanism more stringent, and assigning them a potentially different kinetic rate.

The modeling of the pseudo biological assertions of the previous section may give a wrong intuition about the real life of a modeler. In practice, the collection of what is known about a system evolves rapidly, and models should evolve as new facts are being observed. For instance one may read in a new biology paper the following assertion:

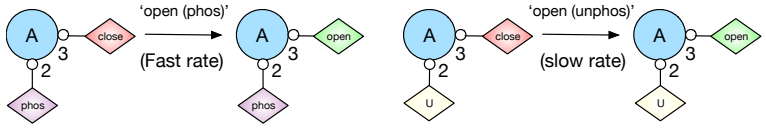
"Protein C may only bind A in its open configuration." (5)

This new observation is simply reflected into the Kappa model by substituting the rule 'A binds C' by:



by conditioning the binding of A to C to the fact that A is in its open form. Now something interesting is happening: before this new (grounded) fact, we chose to hypothesize a refinement of the 'A phos C' rule, requiring A to be phosphorylated, in order to correlate the kinase activity of A with the presence of B. However the new fact suggests

another way of realizing assertion (4): instead of refining rule ‘A phos C’ one simply states that phosphorylation pushes A toward the open configuration, for instance by increasing the rate of the ‘open’ rule when A is phosphorylated:



Doing so allows one to obtain an alternative model that also realizes the assertion (see Figure 12), arguably in a more elegant manner.

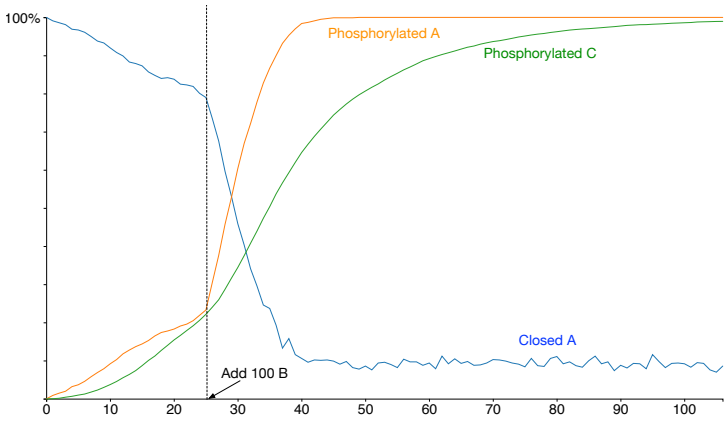


Fig. 12. Simulation of the model incorporating the new observation that only open A can bind C, and the hypothesis that A phosphorylation opens A. The conjunction of the new fact and this hypothesis allows one to realize the assertion “Kinase B activates A”.

3.4. Discussion

An assertion generally needs to be realized by several rules: the more complex the assertion, the more alternatives there are to represent it. For instance an assertion of the form “A can form a stable heterodimer with B” is immediately interpreted as a rule, almost unambiguously; by reference to Figure 6, we say that such assertion is *mechanistic*. An assertion of the form “Kinase A stabilizes the complex C” offers little hint by itself that would guide the modeler in her mechanistic interpretation of this fact. We call these assertions *functional*. We suggested a simple strategy to treat these assertions, based on rule refinements:

(1) Use rules to represent mechanistic assertions. Those rules constitute the kernel of the model.
(2) Verify whether or not functional assertions are already satisfied by the existing rules.
(3) For all assertions that are not satisfied, use rule refinements to induce the missing correlations.

Once some biological facts have been implemented as a collection of Kappa rules, there is still a lot of engineering work to do. Building a model as a program requires the assistance of an IDE and classical tools for programming languages. Some have been implemented for Kappa, such as reachability [Feret 2007; Feret and Lý 2016]

(which is used to infer model types such as the one given in Figure 9.**B**) or causality analysis [Danos et al. 2012]. The latter is particularly useful as causality analysis produces a partial order for rule applications, the composition of which leads to the formation of particular graph observables. This type of analysis can be used for debugging models, to help understand why a particular graph is reached when it should not be, or to produce an explanation of biological functions, in which case they can be used to *explain* why an assertion holds (see Figure 13).

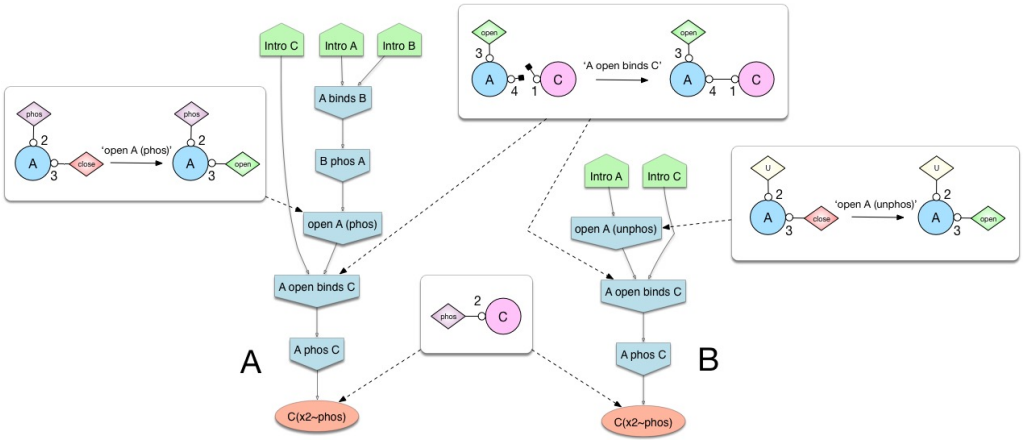


Fig. 13. Causality analysis of our ABC model: asking the simulator to explain C phosphorylation. **A** and **B** are two possible causal sequences of rules leading to the phosphorylation of C, obtained by simulation trace analysis. The relative frequency of **A** and **B** is arbitrated by how likely it is that A opens without being phosphorylated, in which case the **B** pathway (which is independent of kinase B) may unfold. Therefore, under the assumption that the rate of 'open A (phos)' is greater than the rate of 'open A (unphos)', the pathway **A** can be viewed as a derivation tree of the theorem "B activates A".

4. TOWARD A SEMANTICS REVOLUTION IN SYSTEMS BIOLOGY?

How can one evaluate the value of a model? If the question is asked of a biologist, not too hostile to the very notion of modeling, she probably might answer that a model is useful if it is predictive. Because the contrapositive of this expectation would hinder the feasability of building large models, we advocate a different stance. A model is *correct* if it is grounded by biological assertions (observations). A model is *complete* if it is grounded by *all* biological assertions. Said another way, a model is predictive if it is complete. This implies that effort should be put in formally grounding models into biological knowledge and delay the need to fit data at all cost if assertions can be verified by general trends.

We anticipate in this column an evolution of modeling in systems biology that will take the analogy between theorem proving and assertion modeling seriously (see Figure 14). To do this, one needs a formalism that captures protein interactions without forcing the modeler to over interpret experimental results. As we have seen with our running example, interpreting an assertion directly into Kappa forces the modeler to "invent" mechanistic explanations when the assertion is too functional, although one can conceive of strategies to propose new rules in a principled manner. At the level of Kappa there is a confusion between a set of rules that represents some mechanistic assertions (biological axioms) and a set of rules that has been hypothesized in order

to implement a functional assertion (a biological theorem). This missing formalism, which we conjecture should be logical, should play the role of a biological *type* the inhabitant of which should be executable rules *à la* Kappa.

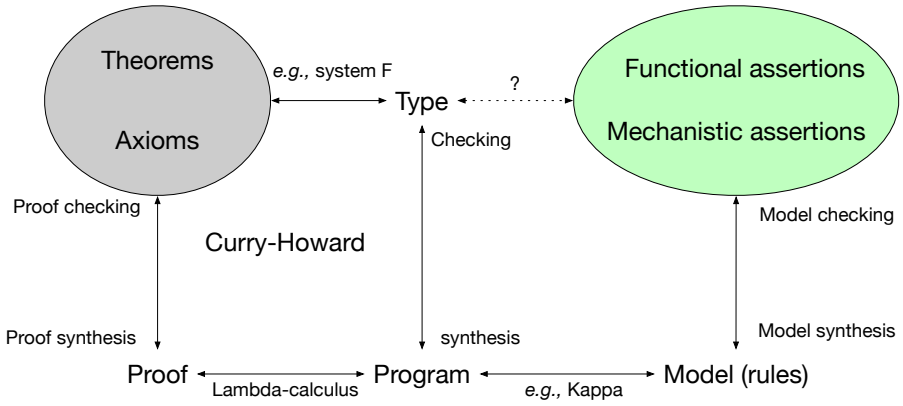


Fig. 14. The missing formalism that would interpret biological assertions. Importantly the distinction between functional and mechanistic assertions depends on the granularity at which one wishes to describe biology. It is partly forced by the choice of language in which model rules are written (here Kappa).

Current conclusion. Physics has brought a cultural revolution in systems biology that is now well integrated in the genome of the discipline. Computers and algorithmics have also changed the daily life of biologists although not in an equally well integrated fashion: many labs still call for the "bioinformatics guy" when needed, although one might expect the new generation of biologists to be better trained in computer science. Beyond computing power and computing techniques, we believe that a fundamental revolution is still to come if one believes that systems biology is not doomed to fail, *e.g.,* in explaining cancer. If combinatorial complexity in *computation* (protein folding, DNA sequence alignment, philogenetics, ...) is the battleground for algorithmics, combinatorial *descriptions* of biological phenomena should be the realm of semantics. We believe that biologists will have to give up relying solely on their intuition to explain how the pathways of the cell work. The "publish or perish world" in which biologists live[8] is not going to let them take this leap of faith into formal models, until semantics provides tools and concepts to unlock previously inaccessible results.

ACKNOWLEDGMENTS

The development of this work is sponsored by the ANR grant ICEBERG 10-BINF-0006 and the Defense Advanced Research Projects Agency (DARPA) and the U. S. Army Research Office under grant number W911NF-14-1-0367. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

REFERENCES

O. Andrei and H. Kirchner. 2007. Graph Rewriting and Strategies for Modeling Biochemical Networks. In *Ninth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2007)*. 407–414. DOI:http://dx.doi.org/10.1109/SYNASC.2007.44

[8]In a more acute manner than most other discipline since experiments cost money, which require fundings, which require Nature papers.

Mehmet Alper Arslan, Ozgur Kutuk, and Huveyda Basaga. 2006. Protein Kinases as Drug Targets in Cancer. *Current Cancer Drug Targets* 6, 7 (2006), 623 – 634.

Adrien Basso-Blandin, Walter Fontana, and Russ Harmer. 2016. A knowledge representation meta-model for rule-based modelling of signalling networks. In Proceedings of the Eleventh International Workshop on *Developments in Computational Models,* Cali, Colombia, October 28, 2015 *(Electronic Proceedings in Theoretical Computer Science)*, César A. Muñoz and Jorge A. Pérez (Eds.), Vol. 204. Open Publishing Association, 47–59. DOI: http://dx.doi.org/10.4204/EPTCS.204.5

Pierre Boutillier, Thomas Ehrhard, and Jean Krivine. 2017a. *Incremental Update for Graph Rewriting*. Springer Berlin Heidelberg, Berlin, Heidelberg, 201–228.

Pierre Boutillier, Jérôme Feret, Jean Krivine, and Kim Quyên Lý. 2017b. KaSim user manual – http://dev.executableknowledge.org/docs/KaSim-manual-master/KaSim_manual.htm. (2017).

Nathalie Chabrier and François Fages. 2003. Symbolic model checking of biochemical networks. In *Proc. CMSB'03 (LNCS)*, Vol. 2602. 146–162.

Saikat Chowdhury and Ram Rup Sarkar. 2015. Comparison of human cell signaling pathway databases—evolution, drawbacks and challenges. *Database: The Journal of Biological Databases and Curation* 2015 (2015), bau126. DOI: http://dx.doi.org/10.1093/database/bau126

F. Ciocchetta and J. Hillston. 2009. Bio-PEPA: a Framework for the Modelling and Analysis of Biochemical Networks. *TCS* 410, 33-34 (2009), 3056–84.

Jean-Paul Comet, Mathilde Noual, Adrien Richard, Julio Aracena, Laurence Calzone, Jacques Demongeot, Marcelle Kaufman, Aurélien Naldi, El Houssine Snoussi, and Denis Thieffry. 2013. On Circuit Functionality in Boolean Networks. *Bulletin of Mathematical Biology* 75, 6 (2013), 906–919.

Troels C. Damgaard, Espen Højsgaard, and Jean Krivine. 2012. Formal Cellular Machinery. *Electronic Notes in Theoretical Computer Science* 284 (2012), 55 – 74. Proceedings of the 2nd International Workshop on Static Analysis and Systems Biology (SASB 2011).

Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, Jonathan Hayman, Jean Krivine, Christopher D. Thompson-Walsh, and Glynn Winskel. 2012. Graphs, Rewriting and Pathway Reconstruction for Rule-Based Models. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*. 276–288.

Vincent Danos, Jérome Féret, Walter Fontana, and Jean Krivine. 2007. Scalable simulation of cellular signaling networks. In *Proc. APLAS'07 (LNCS)*, Vol. 4807. 139–157.

Vincent Danos, Jérôme Feret, Walter Fontanta, Russ Harmer, and Jean Krivine. 2007. Rule based modeling of biological signaling. In *Proceedings of CONCUR 2007 (LNCS)*, Luís Caires and Vasco Thudichum Vasconcelos (Eds.), Vol. 4703. Springer, 17–41.

Vincent Danos, Russ Harmer, and Glynn Winskel. 2013. Constraining rule-based dynamics with types. *Mathematical Structures in Computer Science* 23, 2 (2013), 272–289.

Vincent Danos and Jean Krivine. 2003. Formal molecular biology done in CCS-R. In *Bio-Concur'03, satellite workshop of CONCUR'03 (ENTCS)*, Vol. 180. 31–49.

Vincent Danos and Cosimo Laneve. 2003. Graphs for Formal Molecular Biology. In *Proc. CMSB'03 (LNCS)*, Vol. 2602. 34–46.

James R. Faeder, Mickael L. Blinov, and William S. Hlavacek. 2004. Rule Based modeling of biochemical networks. *Complexity* 10, 4 (2004), 22–41.

Jérôme Feret. 2007. Reachability Analysis of Biological Signalling Pathways by Abstract Interpretation. In *Proceedings of the International Conference of Computational Methods in Sciences and Engineering, ICCMSE'2007, Corfu, Greece (American Institute of Physics Conference Proceedings)*. American Institute of Physics, Corfu, Greece, 619–622.

Jérôme Feret and Kim Quyên Lý. 2016. Reachability analysis via orthogonal sets of patterns.. In *Seventeenth International Workshop on Static Analysis and Systems Biology (SASB'16) (ENTCS)*. elsevier. to appear.

Jasmin Fisher and Thomas A Henzinger. 2007. Executable cell biology. *Nature Biotech* 25, 11 (2007), 1239–1249.

Daniel T. Gillespie. 1976. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *J. Comput. Phys.* 22, 4 (1976), 403–434.

William S Hlavacek, James R Faeder, Michael L Blinov, Richard G Posner, Michael Hucka, and Walter Fontana. 2006. Rules for modeling signal-transduction systems. *Science Signaling* 2006, 344 (2006), re6.

Aileen Houston and Joe O'Connell. 2004. The Fas signalling pathway and its role in the pathogenesis of cancer. *Current Opinion in Pharmacology* 4, 4 (2004), 321–326. DOI: http://dx.doi.org/10.1016/j.coph.2004.03.008

Bernardo A. Huberman and Lada A. Adamic. 1999. Internet: Growth dynamics of the World-Wide Web. *Nature* 401, 6749 (09 09 1999), 131–131. http://dx.doi.org/10.1038/43604

Mathias John, Cédric Lhoussaine, Joachim Niehren, and Cristian Versari. 2011. Biochemical Reaction Rules with Constraints. In *Proc. ESOP 2011 (LNCS)*, Vol. 6602. 338–357.

Stuart Kauffman, Carsten Peterson, Björn Samuelsson, and Carl Troein. 2003. Random Boolean network models and the yeast transcriptional network. *Proceedings of the National Academy of Sciences* 100, 25 (2003), 14796–14799. DOI:http://dx.doi.org/10.1073/pnas.2036429100

Jean Krivine, Robin Milner, and Angelo Troina. 2008. Stochastic Bigraphs. *Electronic Notes in Theoretical Computer Science* 218 (2008), 73 – 96. Proceedings of the 24th Conference on the Mathematical Foundations of Programming Semantics (MFPS XXIV).

Carlos F Lopez, Jeremy L Muhlich, John A Bachman, and Peter K Sorger. 2013. Programming biological models in Python using PySB. *Molecular Systems Biology* 9, 1 (2013). DOI:http://dx.doi.org/10.1038/msb.2013.1

Kanae Oda, Yukiko Matsuoka, Akira Funahashi, and Hiroaki Kitano. 2005. A comprehensive pathway map of epidermal growth factor receptor signaling. *Molecular Systems Biology* 1 (2005), 2005.0010–2005.0010. DOI:http://dx.doi.org/10.1038/msb4100014

Loïc Paulevé and Adrien Richard. 2012. Static Analysis of Boolean Networks Based on Interaction Graphs: A Survey. *ENTCS* 284 (2012), 93 – 104. Proceedings of the 2nd International Workshop on Static Analysis and Systems Biology (SASB 2011).

Michael Pedersen and Gordon Plotkin. 2008. *A Language for Biochemical Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 63–82. DOI:http://dx.doi.org/10.1007/978-3-540-88562-7_9

Andrew Phillips and Luca Cardelli. 2007. Efficient, Correct Simulation of Biological Processes in the Stochastic pi-calculus. In *CMSB 2007*. to appear.

Gheorghe Păun and Francisco J. Romero-Campero. 2008. Membrane computing as a modeling framework. Cellular systems case studies. In *Formal Methods for Computational Systems Biology (LNCS)*, Vol. 5016. 168–214.

J.C Raoult. 1984. On graph rewriting. *TCS* 32 (1984), 1–24.

Aviv Regev and Ehud Shapiro. 2002. Cells as computation. *Nature* 419 (September 2002), 343.

P. Ruet. 2015. Negative local feedbacks in Boolean networks. *ArXiv e-prints* (2015).

René Thomas. 1973. Boolean Formalization of Genetic Control Circuits. *Journal of Theoretical Biology* 42 (1973), 563–585.

Peter J. M. Van Haastert and Peter N. Devreotes. 2004. Chemotaxis: signalling the way forward. *Nat Rev Mol Cell Biol* 5, 8 (08 2004), 626–634. http://dx.doi.org/10.1038/nrm1435

Jia You. 2015. DARPA sets out to automate research. *Science* 347, 6221 (2015), 465.

Conference Reports

JORGE A PÉREZ, University of Groningen, The Netherlands
j.a.perez@rug.nl

In this installment of the conference report column, Andrej Bauer (University of Ljubljana, Slovenia) reports on the 33rd Conference on the Mathematical Foundations of Programming Semantics (MFPS XXXIII) and the 7th Conference on Algebra and Coalgebra in Computer Science (CALCO 2017).

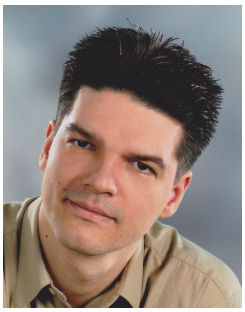
As Andrej explains in his report, MFPS and CALCO are two "cousin conferences" with many points in common; this year, they were co-located in Ljubljana (Slovenia) on June 12-16, 2017.[1] The scientific program of the two conferences was complemented by several tutorials, special sessions, and other satellite events (such as the CALCO workshops on "Early Ideas" and "Tools"). I am most grateful to Andrej for his availability and insightful report.

I look forward to receiving your personal impressions and/or reports on conferences and meetings broadly related to SIGLOG. As usual, I will also be pleased to hear all your ideas and suggestions for future installments of the column.

[1] See http://coalg.org/mfps-calco2017/.

Report on MFPS XXXIII and CALCO 2017

Andrej Bauer, University of Ljubljana, Slovenia



I once heard it said that cousins can be best friends because they understand the quirks of each other's families but are separated enough not to be hindered by sibling rivalry. If this is true, then Mathematical Foundations of Programming Semantics (MFPS) and the Conference on Algebra and Coalgebra in Computer Science (CALCO) are cousins. When it was decided that they would meet again, this time in Ljubljana, Slovenia, one could only expect an enjoyable and productive week in one of the greenest capitals of Europe. The preparations proceeded smoothly under the MFPS program chair Alexandra Silva (University College London), the CALCO program cochairs Barbara König (University of Duisburg-Essen) and Filippo Bonchi (École normale supérieure de Lyon), and the head local organizer Matija Pretnar (University of Ljubljana).

The events kicked off with MFPS on Monday, June 12th 2017, and finished with CALCO on Friday. There was a shared session on Wednesday morning, and on Tuesday there was a separate meeting of Reliable and Privacy-Aware Software Systems (REPAS), a research project funded by the French National Research Agency. The venue was the Faculty of Mathematics and Physics, University of Ljubljana. The new mathematics building offered all that was required, as well as a temporarily malfunctioning cooling system that made everyone feel a very warm welcome.

The MFPS programme was anchored by plenary talks. These were given by Laura Kovács (Vienna Technical University) who spoke about algebraic reasoning for program analysis, Dexter Kozen (Cornell University) who spoke about Borel coalgebra and non-wellfounded logic, and Amr Sabry (Indiana University) who spoke about connections between reversible programs and univalent universes. The fourth plenary speaker, Rehana Patel (Olin College), unfortunately could not attend the meeting, as she found it impossible to pass through the invisible walls that separate countries in today's world. This year MFPS honored Dexter Kozen with a lovely birthday song and an impressive chocolate cake. It would be too crude to measure age in numbers, so let us be content with noting that in the case of Professor Kozen the official number would better reflect reality if it progressed more slowly. In addition to the plenary and contributed talks, MFPS always has several special sessions that focus on particular topics. Each session starts with an invited tutorial, which is followed by several invited talks that flesh out the session and give everyone the opportunity to learn about the latest developments. This year the topics were Algebraic automata theory (headed with a tutorial by Laure Daviaud from Warsaw University), Foundations of network programming (Nate Foster from Cornell University, currently on sabbatical at Barefoot Networks), Concurrent program verification (Derek Dreyer from Max Planck Institute for Software Systems), and Metrics and privacy (headed by James Worrell from Oxford University). The session on metrics in privacy was a joint one with CALCO.

CALCO invited two additional speakers: Alex Simpson (University of Ljubljana) spoke about probability sheaves, and Nicoletta Sabadini (University of Insubria) about

an algebra for open, reconfigurable nets. In addition to the contributed talks CALCO also has a subsidiary Early ideas workshop in which the speakers get a chance to briefly present fresh new ideas that they are still working on, and a Tools workshop for showcasing software tools based on algebraic and coalgbraic methods. The CALCO program committee selected the best paper, which was "Being Van Kampen is a Uniqueness Property in Presheaf Topoi" by Harald König (University of Applied Sciences Hannover) and Uwe Wolter (University of Bergen). The audience voted on the best presentation. The competition was fierce and the knives were out when David Reutter and Jamie Vicary (Oxford University) gave the audience a pair of qubits to play with, and invited everyone to a party at their place. It was revealed that the qubits were simulated by classical circuits, but the party was a huge success, as was evident from the conference attendance the next morning. In the end, the best presentation award went to Fabio Zanasi (University College London) who spoke about "A Universal Construction for (Co)relations".

The contributed talks at MFPS and CALCO spanned a wide spectrum of topics that is difficult to summarize succinctly. At MFPS we could observe an emphasis on probabilistic and quantum computation, although there was no lack of more classical themes such as denotational and game semantics, while CALCO mostly stayed faithful to its title with contributed talks about coalgebras, bisimulation, automata theory, and coalgebraic logic. Here are some dry numbers, for those like them: of 36 MFPS submissions 16 were accepted, yielding an acceptance rate of 44%; CALCO had 38 submissions of which 22 were accepted as regular papers, with a 58% acceptance rate. There were also 11 Early Ideas presentations and one CALCO Tools talk. That is a fairly low number of talks about actual software tools, but I am happy to report that the MFPS special session on concurrent program verification featured a demonstration of Iris, a tool for reasoning about safety of concurrent programs, and several speakers told us that their results were formalized and verified by proof assistants, a welcome trend which hopefully will become even more common in the future.

On Wednesday afternoon, after the last talk, a guided tour of the city of Ljubljana was organized. Two hours of walking around the city and learning everything about the ancient Roman city of Emona, centuries of Hapsburg hegemony, the architectural renaissance under the influence of the Vienna Secession, and the turbulent events of the 20th century, were more than enough to make everyone hungry and thirsty. The dinner was served at the Ljubljana castle, on top of a hill overseeing the city. When the local organization was declared perfect during the obligatory toasts, the participants took a moment from enjoying the local cuisine and wine to support the claim with a loud applause. Indeed, one can hardly be dissatisfied with local organizers that tell the participants to disregard the law[1] and serve ice cream with morning coffee.

What are we to take away from the meeting, apart from heads full of new ideas? That two international conferences with participants from 16 countries who interwove science and friendship, organized in Slovenia, financially supported by EU and USA, and chaired by a Portuguese from the United Kingdom, an Italian from France, and a German from Germany, represent the kind of connected world that we, the scientists, want to live in and will fight for, so that all people may benefit from the fruits of science.

[1] Lest someone take this the wrong way: in parallel with MFPS and CALCO the Faculty of Mathematics and Physics organized the Linear Algebra Workshop (LAW). At some point the participants were in danger of joining LAW, hence the warning.

CFP

SIGLOG Monthly 194

DANIELA PETRIŞAN, Université Paris Diderot

SIGLOG Monthly 194
July 4, 2017

****
* Past issues of the newsletter are available at
http://lii.rwth-aachen.de/lics/newsletters/
* Instructions for submitting an announcement to the newsletter
can be found at
http://lii.rwth-aachen.de/lics/newsletters/inst.html
****

TABLE OF CONTENTS

WINNERS OF THE 2017 CHRUCH AWARD

http://siglog.org/winners-of-the-2017-alonzo-church-award/

* The 2017 Alonzo Church Award for Outstanding Contributions to Logic and Computation is given jointly to Samson Abramsky, Radha Jagadeesan, Pasquale Malacaria, Martin Hyland, Luke Ong, and Hanno Nickau for providing a fully-abstract semantics for higher-order computation through the introduction of game models, thereby fundamentally revolutionising the field of programming language semantics, and for the applied impact of these models.
* Their contributions appeared in three papers:
  - S. Abramsky, R. Jagadeesan, and P. Malacaria. Full Abstraction for PCF. Information and Computation, Vol. 163, No. 2, pp. 409 - 470, 2000.
  - J.M.E. Hyland and C.-H.L. Ong. On Full Abstraction for PCF: I, II, and III. Information and Computation, Vol. 163, No. 2, pp. 285 - 408, 2000.
  - H. Nickau. Hereditarily sequential functionals. Proc. Symp. Logical Foundations of Computer Science: Logic at St. Petersburg (eds. A. Nerode and Yu.V. Matiyasevich), Lecture Notes in Computer Science, Vol. 813, pp. 253 - 264. Springer-Verlag, 1994.

  A description of the contributions is available at
  http://siglog.org/winners-of-the-2017-alonzo-church-award/
* The 2017 award will be presented at the 26th Computer Science Logic (CSL) Conference, the annual meeting of the European Association for Computer Science Logic. This will be held August 20th - 24th, 2017, at Stockholm University, Sweden.

THE 2018 FEDERATED LOGIC CONFERENCE (FLoC 2018)

Preliminary annoucement
6-19 July 2018
Oxford, England UK
http://www.floc2018.org/

* In 1996, as part of its Special Year on Logic and Algorithms, DIMACS hosted the first Federated Logic Conference (FLoC). It was modelled after the successful Federated Computer Research Conference (FCRC), and synergetically brought together conferences that apply logic to computer science.
* We are pleased to announce the seventh Federated Logic Conference (FLoC'18) to be held in Oxford, UK, in July 2018, at the Mathematical Institute and the Blavatnik School of Government at the University of Oxford.
* FLoC 2018 brings together nine major international conferences related to mathematical logic and computer science:
  International Conference on Computer Aided Verification (CAV)
  IEEE Computer Security Foundations Symposium (CSF)
  International Symposium on Formal Methods (FM)
  International Conference on Formal Structures for Computation and Deduction (FSCD)
  International Conference on Logic Programming (ICLP)
  International Joint Conference on Automated Reasoning (IJCAR)
  International Conference on Interactive Theorem Proving (ITP)
  Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)

International Conference on Theory and Applications of Satisfiability Testing (SAT)
Plus FLoC workshops (7-8 July, 13 July, and 18-19 July) and the School on Foundations of Programming and Software Systems (FoPSS, 30 June - 6 July).

* We have already begun confirming exciting lineup of speakers, including keynotes by Shafi Goldwasser and Georges Gonthier; plenary lectures by Peter O'Hearn and Byron Cook; and a public lecture by Stuart Russell at the Sheldonian Theatre. We will also hold an Oxford Union-style debate on ethics for autonomous robots. There will be banquets, receptions and other social events in historic venues across the city: see www.floc2018.org/social-events/ for the latest updates.
* PROGRAMME COMMITTEE
General Chair: Moshe Y. Vardi
Co-chairs: Daniel Kroening, Marta Kwiatkowska
CAV: Hana Chockler
CSF: Stephen Chong
FM: Bill Roscoe
FSCD: Hane Kirchner
ICLP: Paul Tarau
IJCAR: Roberto Sebastiani
ITP: Assia Mahboubi
LICS: Martin Hofmann
SAT: Olaf Byersdorff

ACM SIGLOG ANNOUNCEMENT
http://siglog.acm.org

* The ACM has recently chartered a Special Interest Group on Logic and Computation (ACM SIGLOG).
* We are pleased to announce the 2016 ACM SIGLOG election results for the term of 1 July 2016 - 30 June 2019. The SIGLOG Chair is Prakash Panangaden and the other officers are Luke Ong (vice-Chair), Amy Felty (Treasurer) and Alexandra Silva (Secretary).
* The ACM-IEEE Symposium on Logic in Computer Science is the flagship conference of SIGLOG. SIGLOG will also actively seek association agreements with other conferences in the field. A SIGLOG newsletter (SIGLOG News) is also published quarterly in an electronic format with community news, technical columns, members' feedback, conference reports, book reviews and other items of interest to the community.
* One can join SIGLOG by visiting
https://campus.acm.org/public/qj/gensigqj/siglist/gensigqj_siglist.cfm
It is possible to join SIGLOG without joining ACM (the SIGLOG membership fee is $25 and $15 for students).

EATCS Bulletin - Call for abstracts

* EATCS Bulletin, http://eatcs.org/index.php/eatcs-bulletin, has a section for "Technical contributions." To stimulate this section further, we will be considering for publication abstracts of works

that have been accepted by journals and/or conferences or have appeared in major archives. The topics of interest include all areas of theoretical computer science (for instance, see topics of the three Tracks of ICALP, http://www.easyconferences.eu/icalp2016/cfp.html).
* Abstracts should be rather detailed, 2-3 pages long in the format given at http://eatcs.org/index.php/eatcs-bulletin. Submissions should include the information on the full paper (the name of conferences, archives, etc) and sufficiently detailed explanation of its merits, e.g., importance, motivations, clear comparison with existing results and novelty and/or new ideas of proof techniques.
* The Bulletin is published in Feb, Jun and Oct. The deadline for the abstract submission is 20th of the previous month, for instance, May 20 (Sat) 2017 for the June issue of this year. All materials including tex and pdf files should be sent electrically to bulletin@eatcs.org and iwama@kuis.kyoto-u.ac.jp. Acceptance/rejection, decided based on its merit mentioned above, will be notified as soon as possible. The Bulletin will not require copy-right transfer for accepted abstracts.

DATES
* RV-CuBES
Call for contributions
Deadline extension: abstracts - July 8, final submission - July 15
http://rv2017.cs.manchester.ac.uk/rv-cubes/
affiliated with RV 2017
* HELMUTH VEITH STIPEND
Call for Applications
http://bit.ly/Forsyte-Helmut-Veith-Stipend
Deadline: August 20, 2017
* PODS 2018
Call for Papers
June 11 - June 13, 2018, Houston, Texas, USA
PODS has two rounds of submissions (see dates below).
Paper submission (2nd cycle): Dec 19, 2017
* ISSTA & SPIN 2017
Call for Participation
July 10-14, 2017, Santa Barbara, California, USA
http://conf.researchr.org/home/issta-2017
http://conf.researchr.org/home/spin-2017
* CAV 2017
Call for Participation
Heidelberg, Germany, 22-28 July 2017
http://www.cavconference.org/2017
* FSTTCS 2017
Call for Papers
Kanpur, India, December 11 - 15, 2017
Conference website: http://fsttcs.org/
Submission deadline: July 24
* CCA 2017
Call for Participation

July 24-27, 2017, Daejeon, South Korea
http://cca-net.de/cca2017/

* RW 2017
Call for Applications- PARTICIPATION TOKENS STILL AVAILABLE
The 13th Reasoning Web Summer School (RW 2017)
London, U.K., July 7-11, 2017
http://reasoningweb.org/2017

* EPS 2017
Call for Posters and Encyclopedia Entries
24, 25th of September 2017, Brasilia, Brazil
http://proofsystem.github.io/Encyclopedia/
Submission: August 1

* TPNC 2017
Call for papers
Prague, Czech Republic, December 18-20, 2017
Faculty of Mathematics and Physics, Charles University
http://grammars.grlmc.com/TPNC2017/
Paper submission: August 6, 2017

* CSL 2017
Call for Participation
August 20 - 24, 2017, Stockholm, Sweden
http://logic.math.su.se/csl-2017

* LOGIC AND AUTOMATA THEORY
Call for participation
A one-day workshop in memory of Zoltan Esik
A satellite event of CSL 2017
Stockholm, August 25, 2017
https://www.imsc.res.in/~jam/esik/zoltan.html

* CPP 2018
Call for Papers
January 8-9, 2018, Los Angeles, USA
http://popl18.sigplan.org/track/CPP-2018
Full paper submission deadline: Wed 11 Oct 2017

* FoIKS 2018
Call for papers
May 14-18, 2018, Alfred Renyi Institute of Mathematics, Budapest, Hungary
http://2018.foiks.org/
Paper submission: December 01, 2017

AN INTERNATIONAL WORKSHOP ON COMPETITIONS, USABILITY, BENCHMARKS, EVALUATION, AND STANDARDISATION FOR RUNTIME VERIFICATION TOOLS (RV-CuBES 2017)
Call for contributions
http://rv2017.cs.manchester.ac.uk/rv-cubes/
affiliated with RV 2017

* HIGHLIGHTS
The goal of this workshop is to provide a venue to discuss ongoing efforts to improve how we evaluate and compare tools for runtime verification. We invite two kinds of submissions: tool overview papers of existing tools and position papers. Attendance at the workshop is not compulsory for submission, but encouraged The

workshop will be integrated into RV 2017 to engage with the wider RV community

* RELATION TO RV 2017

There is no overlap in scope between this workshop and RV 2017. Any papers containing original technical developments should be submitted to RV 2017 rather than this workshop as the workshop focuses on tool reviews (containing existing work) and position statements. If there is any uncertainty please contact the workshop chairs. The workshop will be integrated into RV 2017.

* SUBMISSIONS

We invite two forms of submission: Tool Overview papers and Position papers. All submissions will be subject to a lightweight review by the PC to ensure a reasonable standard and to provide constructive feedback to improve the quality of the submission.

* IMPORTANT DATES

Abstracts: 8 July 2017
Final Submission: 15 July 2017
Notification 1 August 2017
Workshop at RV 2017 13-16 September 2017
Post-proceedings deadline 14 October 2017

* Program Committee Chairs

Giles Reger, University of Manchester, UK
Klaus Havelund, NASA Jet Propulsion Laboratory, USA

HELMUTH VEITH STIPEND

Call for Applications
Deadline: August 20, 2017

* It is our pleasure to invite academically excellent female applicants in pursuit (or planning to pursue) a master degree in Computer Sciences at TU Wien to apply for Helmut Veith Stipend. The recipients of Helmut Veith Stipend receive EUR 6000 annually for the duration of up to two years, and waiver of all tuition fees for the study at TU Wien. The application deadline is August 20, 2017. For more information download the Flyer for Helmut Veith Stipend.

* ABOUT HELMUT VEITH

The stipend is named to honour the memory of Helmut Veith (1971 - 2016), specifically his international influential research, and his visionary mentorship of building bridges between the computer science and the society. The Helmut Veith Stipend continues the late scholar's support of female scientist in the field of computer science, and his backing of the revival era in the Austrian logic scene.

* More information here

http://bit.ly/Forsyte-Helmut-Veith-Stipend

37th ACM SIGMOD-SIGACT-SIGAI Symposium on PRINCIPLES OF DATABASE SYSTEMS (PODS 2018)

Call for Papers
June 11 - June 13, 2018, Houston, Texas, USA
PODS has two rounds of submissions (see dates below).

* The PODS symposium series, held in conjunction with the SIGMOD conference series, provides a premier annual forum for the communication of new advances in the theoretical foundations of data management, traditional or non-traditional (see http://www.sigmod.org/the-pods-pages/the-pods-pages).
* For the 37th edition, PODS continues to aim to broaden its scope, and calls for research papers providing original, substantial contributions along one or more of the following aspects:
- deep theoretical exploration of topical areas central to data management;
- new formal frameworks that aim at providing the basis for deeper theoretical investigation of important emerging issues in data management;
- validation of established theoretical approaches from the lens of practical applicability in data management. Papers in this track should provide an experimental evaluation that gives new insight in established theories. Besides, they should provide a clear message to the database theory community as to which aspects need further (theoretical) investigation, based on the experimental findings.
* Topics (that fit the interests of the symposium include, but are not limited to):
- concurrency & recovery, distributed/parallel databases, cloud computing
- data and knowledge integration and exchange, data provenance, views and data warehouses, metadata management
- data-centric (business) process management, workflows, web services
- data management and machine learning
- data mining, information extraction, search
- data models, data structures, algorithms for data management
- data privacy and security, human-related data and ethics
- data streams
- design, semantics, query languages
- domain-specific databases (multi-media, scientific, spatial, temporal, text)
- graph databases and (semantic) Web data
- incompleteness, inconsistency, uncertainty in data management
- knowledge-enriched data management
- model theory, logics, algebras, computational complexity
* PROGRAM CHAIR: Marcelo Arenas
* IMPORTANT DATES:
First Submission Cycle:
- Abstract submission: Jun 15, 2017
- Paper submission: Jun 22, 2017
- First notification: Aug 31, 2017
- Revision deadline: Sep 28, 2017
- Final notification: Nov 02, 2017
Second Submission Cycle:
- Abstract submission: Dec 12, 2017
- Paper submission: Dec 19, 2017
- Final notification: Feb 27, 2018
All deadlines end at 11:59pm AoE.

26TH ACM SIGSOFT INTERNATIONAL SYMPOSIUM ON SOFTWARE TESTING AND ANALYSIS
24TH INTERNATIONAL SPIN SYMPOSIUM ON MODEL CHECKING OF SOFTWARE
(ISSTA & SPIN 2017)
July 10-14, 2017, Santa Barbara, California, USA
http://conf.researchr.org/home/issta-2017
http://conf.researchr.org/home/spin-2017
* ISSTA is the leading research symposium on software testing and analysis, bringing together academics, industrial researchers, and practitioners to exchange new ideas, problems, and experiences on how to analyze and test software systems.
* The SPIN symposium brings together researchers and practitioners interested in automated, tool-based techniques to analyze software systems and models of software systems for verification and validation purposes.
* RESEARCH PROGRAM
ISSTA list of accepted papers:
http://conf.researchr.org/info/issta-2017/accepted-papers
SPIN list of accepted papers:
http://conf.researchr.org/info/spin-2017/accepted-papers
* KEYNOTE SPEAKERS
ISSTA
Christopher Kruegel, UCSB
Armando Solar-Lezama, MIT
SPIN
Domagoj Babic, Google
Byron Cook, Amazon Web Services
Gerard Holzmann, Nimble Research
* CO-LOCATED EVENTS
ISSTA Doctoral Symposium
http://conf.researchr.org/track/issta-2017/issta-2017-doctoral-symposium
ISSTA Demonstrations track
http://conf.researchr.org/track/issta-2017/issta-2017-demos
TECPS 2017: Workshop on Testing Embedded and Cyber-Physical Systems
http://conf.researchr.org/track/issta-2017/issta-2017-tecps
RERS Challenge 2017: Rigorous Examination of Reactive Systems
http://www.rers-challenge.org/2017/
* FOLLOW US ON SOCIAL MEDIA
http://facebook.com/isstaconf
http://twitter.com/issta_conf
http://twitter.com/hashtag/spin17sym

COMPUTER-AIDED VERIFICATION, 29TH INTERNATIONAL CONFERENCE (CAV 2017)
Call for Participation
Heidelberg, Germany, 22-28 July 2017
http://www.cavconference.org/2017
* TL;DR Early registration: June 14, 2017; Hotel booking deadlines: early June, 2017; We hope to welcome you at CAV -- we have an exciting program!
* ABOUT CAV

CAV 2017 is the 29th in a series dedicated to the advancement of the theory and practice of computer-aided formal analysis methods for hardware and software systems. The conference covers the spectrum from theoretical results to concrete applications, with an emphasis on practical verification tools and the algorithms and techniques that are needed for their implementation. Along with the main conference, CAV will feature eight workshops (including a special workshop in honor of David Dill) and tutorials.

* HIGHLIGHTS:
-- WORKSHOPS (22-23 July)
-- VERIFICATION MENTORING WORKSHOP (23 July)
-- SPECIAL WORKSHOP Dill@60 in Honor of David Dill (24 July)
-- MAIN CONFERENCE (24-28 July)

* INVITED SPEAKERS
Chris Hawblitzel, Microsoft Research
Marta Kwiatkowska, Oxford
Viktor Vafeiadis, MPI-SWS
Winner of the CAV award (to be announced at the conference)

* INVITED TUTORIALS
Loris D'Antoni, University of Wisconsin-Madison: The power of symbolic automata and transducers
Mayur Naik, University of Pennsylvania: Maximum Satisfiability in Software Analysis: Applications and Techniques

* PUBLIC LECTURE "Logic Lounge" in memory of Helmut Veith
Fabiana Zollo: Social Dynamics in the Post-Truth Society: How the Confirmation Bias is Changing the Public Discourse

* SATELLITE EVENTS (22-23 July)
7 satellite workshops will take place before CAV 2017.
Check their calls for papers and consider contributing!
SYNT - Sixth Workshop on Synthesis (July 22)
DARS - Design and Analysis of Robust Systems (July 22)
NSV/Rise4CPS - Numerical Software Verification and Formal Methods for Rigorous Systems Engineering of Cyber-Physical Systems (July 22-23)
SMT - Satisfiability Modulo Theories (July 22-23)
VSTTE - Verified Software: Theories, Tools, and Experiment (July 22-23)
VMW - Verification Mentoring Workshop (July 23)
FEVER - Formal Approaches to Explainable VERification (July 23)

37TH ANNUAL CONFERENCE ON FOUNDATIONS OF SOFTWARE TECHNOLOGY AND THEORETICAL COMPUTER SCIENCE (FSTTCS 2017)

Call for Papers
Kanpur, India, December 11 - 15, 2017
Conference website: http://fsttcs.org/

* TOPICS
Representative areas include, but are not limited to, the following.
- Algorithms and Data Structures
- Algorithmic Graph Theory and Combinatorics
- Approximation Algorithms

- Automata and Formal Languages
- Combinatorial Optimization
- Communication Complexity
- Computational Biology
- Computational Complexity
- Computational Geometry
- Computational Learning Theory
- Cryptography and Security
- Game Theory and Mechanism Design
- Logic in Computer Science
- Model Theory, Modal and Temporal Logics
- Models of Concurrent and Distributed Systems
- Models of Timed, Reactive, Hybrid and Stochastic Systems
- Parallel, Distributed and Online Algorithms
- Parameterized Complexity
- Principles and Semantics of Programming Languages
- Program Analysis and Transformation
- Proof Complexity
- Quantum Computing
- Randomness in Computing
- Specification, Verification, and Synthesis
- Theorem Proving, Decision Procedures, and Model Checking
- Theoretical Aspects of Mobile and High-Performance Computing

* INVITED SPEAKERS

Sham Kakade (University of Washington, USA)
Anca Muscholl (LaBRI & Universite de Bordeaux, France)
Devavrat Shah (MIT, USA)
Vinod Vaikuntanathan (MIT CSAIL, USA)
Thomas Wilke (Christian-Albrechts-Universitaet zu Kiel, Germany)

* IMPORTANT DATES

Paper submission deadline: Monday, July 24, 2017 (Anywhere on earth)
Notification to Authors: Monday, September 18, 2017
Camera-ready deadline: Monday, October 16, 2017
Conference: December 11 - 15, 2017

FOURTEENTH INTERNATIONAL CONFERENCE ON COMPUTABILITY AND COMPLEXITY IN ANALYSIS (CCA 2017)

Call for Participation
July 24-27, 2017, Daejeon, South Korea
http://cca-net.de/cca2017/

* ABOUT. This conference is concerned with the theory of computability and complexity over real-valued data. The topics of interest include foundational work on various models and approaches for describing computability and complexity over the real numbers. They also include complexity-theoretic investigations, both foundational and with respect to concrete problems, and new implementations of exact real arithmetic, as well as further developments of already existing software packages. The conference CCA 2017 is preceded by the 15th Asian Logic Conference from July 10 to 14 and followed by the Workshop on Real Verification on Friday, July 28.

* INVITED SPEAKERS

Hee-Kap Ahn (Pohang, Republic of Korea)
Veronica Becher (Buenos Aires, Argentina)
Anders Hansen (Cambridge, UK)
Takayuki Kihara (Berkeley, USA)
Amaury Pouly (Max Planck Institute, Germany)
Linda Brown Westrick (Connecticut, USA)

* REGISTRATION

Early bird registration is open until June 10th:
http://kaist.kiwii.co.kr/

* SATELLITE WORKSHOP ON REAL VERIFICATION

A co-located "Workshop on Real Verification" will take place on
Friday, July 28
https://complexity.kaist.edu/CCA2017/workshop.html

THE 13TH REASONING WEB SUMMER SCHOOL (RW 2017)

Call for Applications
London, U.K., July 7-11, 2017
http://reasoningweb.org/2017

* co-located with:

- RuleML+RR: International Joint Conference on Rules and Reasoning
  London, U.K., July 12-15, 2017
  http://2017.ruleml-rr.org
- RuleML+RR Doctoral Consortium
  http://2017.ruleml-rr.org/doctoral-consortium/
- DecisionCAMP 2017
  London, U.K., July 13-14, 2017
  http://2017.ruleml-rr.org/decisioncamp-2017
- 11th International Rule Challenge
  London, U.K., July 12-15, 2017
  http://2017.ruleml-rr.org/calls/international-rule-challenge/

* The purpose of the Reasoning Web Summer School is to disseminate recent advances on reasoning techniques which are of particular interest to Semantic Web and Linked Data applications. The school is primarily intended for postgraduate (PhD or MSc) students, postdocs, young researchers, and senior researchers wishing to learn about Reasoning on the Semantic Web and related issues. In 2017, the theme of the school is:

"Semantic Interoperability on the Web"

* IMPORTANT DATES

Application deadline: May 20, 2017 (extended)
Notifications: May 25, 2017
Registration deadline: May 31, 2017

* LECTURES

- Andrea Cali (Birkbeck University of London, U.K.)
  "Ontology querying: Datalog strikes back"
- Thomas Eiter (Technical University of Wien, Austria)
  "Answer Set Programming with External Source Access"
- Thomas Lukasiewicz (University of Oxford, U.K.)
  "Uncertainty Reasoning for the Semantic Web"
- Marco Montali (Free University of Bolzano/Bozen, Italy)
  "Ontology-based Data Access for Log Extraction in Process Mining"

- Axel Polleres (Vienna University of Economics & Business, Austria)
  "Challenges for Semantic Data Integration on the Web of Open Data"
- Marie-Christine Rousset (University Grenoble-Alpes,
  Institut Universitaire de France)
  "Datalog revisited for reasoning in Linked Data"
- Torsten Schaub (University of Potsdam, Germany and
  Inria, Bretagne Atlantique, Rennes, France)
  "A Tutorial on Hybrid Answer Set Solving"
- Juan Sequeda (Capsenta, USA)
  "Integrating Relational Databases with the Semantic Web"
- Giorgos Stamou (National Technical University of Athens, Greece)
  "Ontological query answering over semantic data"

* FURTHER DETAILS on applications, student fees and grants, venue, are available on the school website.

ENCYCLOPEDIA OF PROOF SYSTEMS - POSTER SESSION & TASK-FORCE (EPS 2017)

Call for Posters and Encyclopedia Entries
24, 25th of September 2017, Brasilia, Brazil
http://proofsystem.github.io/Encyclopedia/

* DESCRIPTION. The Encyclopedia of Proof Systems was created in 2014 with the goal of being a quick reference for the various proof systems used by logicians. Since then, it has collected 64 entries on the most various logics and calculi. This was only possible due to the collaboration of many members of the logic community. This event aims to promote the encyclopedia and attract more contributions and collaborators. It consists of:
- a poster session in the afternoon of September 24th, 2017, during which submitted entries will be displayed as posters;
- an interactive hands-on meeting in the morning of September 25th, 2017, for those who would like to contribute to the continuous improvement of the encyclopedia.

* Submissions and instructions are available in the website:
http://proofsystem.github.io/Encyclopedia/

* IMPORTANT DATES
Submission: 1 August 2017
Notification: 15 August 2017

6th INTERNATIONAL CONFERENCE ON THE THEORY AND PRACTICE OF NATURAL COMPUTING (TPNC 2017)

Call for papers
Prague, Czech Republic, December 18-20, 2017
Faculty of Mathematics and Physics, Charles University
http://grammars.grlmc.com/TPNC2017/

* TPNC is a conference series intending to cover the wide spectrum of computational principles, models and techniques inspired by information processing in nature. TPNC 2017 will reserve significant room for young scholars at the beginning of their career and particular focus will be put on methodology. The conference aims at attracting contributions to nature-inspired models of computation, synthesizing nature by means of computation, nature-inspired

materials, and information processing in nature.
* PROGRAMME CHAIR
Carlos Martin-Vide (Rovira i Virgili University, ES, co-chair)
Roman Neruda (Prague, co-chair)
* DEADLINES (all at 23:59 CET):
Paper submission: August 6, 2017
Notification of paper acceptance or rejection: September 6, 2017
Final version of the paper for the LNCS proceedings: September 16, 2017
Early registration: September 16, 2017
Late registration: December 4, 2017
Submission to the post-conference journal special issue: March 20, 2018
* QUESTIONS AND FURTHER INFORMATION:
david.silva409 (at) yahoo.com

26TH EACSL ANNUAL CONFERENCE ON COMPUTER SCIENCE LOGIC (CSL 2017)
Call for Participation
August 20 - 24, 2017, Stockholm, Sweden
http://logic.math.su.se/csl-2017
* Computer Science Logic (CSL) is the annual conference of the European Association for Computer Science Logic (EACSL). It is an interdisciplinary conference, spanning across both basic and application oriented research in mathematical logic and computer science and is intended for computer scientists whose research involves logic, as well as for logicians working on issues essential for computer science.
* CSL 2017 will be co-located with several other logic-related events, taking place at Stockholm University, including the 3rd Nordic Logic Summer School, NLS 2017, August 7-11, and the Logic Colloquium 2017 (LC 2017), August 14-20.
* INVITED SPEAKERS
* Invited highlight speakers for the LC-CSL joint session on August 20:
Veronica Becher (University of Buenos Aires)
Phokion Kolaitis (University of California Santa Cruz and IBM Research - Almaden)
Pierre Simon (UC Berkeley)
Wolfgang Thomas (RWTH Aachen)
* CSL plenary speakers:
Laura KovÃącs (Vienna University of Technology)
Stephan Kreutzer (Technische UniversitÃđt Berlin)
Meena Mahajan (Institute of Mathematical Sciences, Chennai)
Margus Veanes (Microsoft Research)
* SPECIAL AND AFFILIATED EVENTS
In addition to the plenary and contributed talks CSL 2017, the conference will also include the following events:
- Presentation of the Alonzo Church award for Outstanding Contributions to Logic and Computation,
- Presentation of the EACSL Ackermann award for Outstanding Dissertation on Logic in Computer Science,
- CSL-affiliated workshops, to be held as CSL co-located events:
Workshop on Logic and Algorithms in Computational Linguistics

LACompLing'17 (August 16-19)
Workshop on Logical Aspects of Multi-Agent Systems
LAMAS 2017 (August 25)
Workshop on Logic and Automata Theory (in memory
of Zoltan Ezik) (August 25)
* Further information about all events can be found on
http://logic.math.su.se/logic-in-stockholm-2017

LOGIC AND AUTOMATA THEORY
Call for participation
A one-day workshop in memory of Zoltan Esik
A satellite event of CSL 2017
Stockholm, August 25, 2017
https://www.imsc.res.in/~jam/esik/zoltan.html
* SPEAKERS
Mikolaj Bojanczyk (University of Warsaw)
Anna Ingolfsdottir (Reykjavik University)
Szabolcs Ivan (University of Szeged)
Wolfgang Thomas (RWTH, Aachen)
Pascal Weil (LaBRI, CNRS and Univ. of Bordeaux)
* Registration link
https://www.math-stockholm.se/konferenser-och-akti/
logic-in-stockholm-2/26th-eacsl-annual-co/
computer-science-logic-2017-august-20-24-1.717663
* Organisers
R. Ramanujam (jam@imsc.res.in) and
Thomas Schwentick (thomas.schwentick@tu-dortmund.de)

SEVENTH ACM SIGPLAN INTERNATIONAL CONFERENCE ON CERTIFIED PROGRAMS AND PROOFS (CPP 2018)
Call for Papers
January 8-9, 2018, Los Angeles, USA
http://popl18.sigplan.org/track/CPP-2018
* OVERVIEW
Certified Programs and Proofs (CPP) is an international forum on
theoretical and practical topics in all areas, including computer
science, mathematics, and education, that consider certification as
an essential paradigm for their work. Certification here means
formal, mechanized verification of some sort, preferably with
production of independently checkable certificates.
* DATES
- Abstract submission deadline: Fri 6 Oct 2017
- Full paper submission deadline: Wed 11 Oct 2017
- Notification: Tue 14 Nov 2017
* PROGRAM CO-CHAIRS
- June Andronick (Data61, CSIRO and UNSW, Australia)
- Amy Felty (University of Ottawa, Canada)

10th SYMPOSIUM ON FOUNDATIONS OF INFORMATION AND KNOWLEDGE SYSTEMS

(FoIKS 2018)
Call for papers
May 14-18, 2018, Alfred Renyi Institute of Mathematics,
Budapest, Hungary
http://2018.foiks.org/
* FoIKS 2018 solicits original contributions dealing with any foundational aspect of information and knowledge systems. This includes submissions that apply ideas, theories or methods from specific disciplines to information and knowledge systems. Examples of such disciplines are discrete mathematics, logic and algebra, model theory, information theory, complexity theory, algorithmics and computation, statistics and optimization.
* The FoIKS symposia are a forum for intense discussions. Speakers will be given sufficient time to present their ideas and results within the larger context of their research; furthermore, participants will be asked to prepare a first response to another contribution in order to initiate discussion.
* INVITED SPEAKERS
Laura Kovacs (TU Wien),
Sebastian Link (Auckland Univ.),
David Pearce (TU Madrid),
Bernhard Thalheim (Christian-Albrechts Univ. Kiel)
* Scientific Sponsors: ALP, EATCS, Vienna Center for Logic and Algorithms
* IMPORTANT DATES
Abstract submission due: November 24, 2017;
Paper submission: December 01, 2017;
Notification: February 02, 2018
* Detailed information can be found on the webpage: http://2018.foiks.org/

PART-TIME (50%) FACULTY POSITION IN COMPUTER SCIENCE - SOFTWARE LANGUAGES LAB IN BRUSSELS
* The computer science department of the Vrije Universiteit Brussel is offering a part-time position as professor to reinforce its software languages and software engineering branch. The position is published under the heading:
WE/2017/001 - Senior Academic Staff - 50% - Science and Bio-Engineering Sciences - Computer Sciences - Software Language Engineering
on the university's job offers website located at
http://vub.talentfinder.be.
The planned starting date for this position is October 1st, 2017.
Contract duration is 2 academic years.
* The deadline for applying is
July 24th, 2017.
Applications should reach us through the website.
* CONTACT
Viviane Jonckers
Software Languages Lab
email: vejoncke@vub.ac.be
phone: +32 2 629 29 67

* FUNCTION
- Research
The selected candidate has to reinforce the research of the Software Languages Lab (SOFT - http://soft.vub.ac.be) which focusses on 'software language engineering'. The lab is active in programming language research and software engineering research. We are specifically looking for a candidate who can link these two domains, e.g. by designing new language features or language extensions with a specific focus on modern software engineering problems or by studying the software engineering aspects that emerge from new âĂIJnearly mainstreamâĂİ languages such as Scala, Clojure, Julia, etc. The candidate's research methodology can be formalism-based, artefact-driven, or experimental. Candidates whose academic track record shows the ability to combine several methodologies are explicitly welcome.
- Teaching
The teaching assignment may include both general courses on bachelor or master level and more specialized courses on MA level. The exact course list will be negotiated with the candidate and will depend on his/her expertise and language skills.

12 PH.D. STUDENT POSITIONS IN ALGORITHMS, VERIFICATION, AND LOGIC
RWTH Aachen University
Call for Applications
Deadline for applications: July 14, 2017
Starting date positions: October 1, 2017
https://moves.rwth-aachen.de/research/projects/unravel/
* The RWTH Aachen University is looking for enthusiastic and highly qualified doctoral researchers. 12 positions are available within the Research Training Group (RTG) UnRAVeL. The key emphasis of an RTG is on the qualification of doctoral researchers with a focused research program and a structured training strategy. UnRAVeL aims to significantly advance probabilistic modelling and analysis for uncertainty by developing new theories, algorithms, and tool-supported verification techniques, and to apply them to core problems from security, planning, and safety and performance analysis.
* Application procedure, required profile, job description, and the possible Ph.D. projects are all available on the web page.
* Involved supervisors: Martin Grohe, Erich Gradel, Erika Abraham, Jurgen Giesl, Joost-Pieter Katoen, Christof Loding, Britta Peis, Gerhard Woeginger, Gerhard Lakemeyer, Ulrike Meyer, Nils Niessen, Pascal Schweitzer.

join today!

SIGLOG & ACM

siglog.acm.org www.acm.org

The **Special Interest Group on Logic and Computation** is the premier international community for the advancement of logic and computation, and formal methods in computer science, broadly defined.

The **Association for Computing Machinery** (ACM) is an educational and scientific computing society which works to advance computing as a science and a profession. Benefits include subscriptions to *Communications of the ACM*, *MemberNet*, *TechNews* and *CareerNews*, full and unlimited access to online courses and books, discounts on conferences and the option to subscribe to the ACM Digital Library.

- ❑ SIGLOG (ACM Member) ..... $ 25
- ❑ SIGLOG (ACM Student Member & Non-ACM Student Member) ..... $ 15
- ❑ SIGLOG (Non-ACM Member) ..... $ 25
- ❑ ACM Professional Membership ($99) & SIGLOG ($25) $124
- ❑ ACM Professional Membership ($99) & SIGLOG ($25) & ACM Digital Library ($99) $223
- ❑ ACM Student Membership ($19) & SIGLOG ($15) ..... $ 34

payment information

Name _____

ACM Member # _____

Mailing Address _____

_____

City/State/Province _____

ZIP/Postal Code/Country _____

Email _____

Mobile Phone _____

Fax _____

Mailing List Restriction
ACM occasionally makes its mailing list available to computer-related organizations, educational institutions and sister societies. All email addresses remain strictly confidential. Check one of the following if you wish to restrict the use of your name:

- ❑ ACM announcements only
- ❑ ACM and other sister society announcements
- ❑ ACM subscription and renewal notices only

Credit Card Type: ❑ AMEX ❑ VISA ❑ MC

Credit Card # _____

Exp. Date _____

Signature _____

Make check or money order payable to ACM, Inc

ACM accepts U.S. dollars or equivalent in foreign currency. Prices include surface delivery charge. Expedited Air Service, which is a partial air freight delivery service, is available outside North America. Contact ACM for more information.

Questions? Contact:
ACM Headquarters
2 Penn Plaza, Suite 701
New York, NY 10121-0701
voice: 212-626-0500
fax: 212-944-1318
email: acmhelp@acm.org

Remit to:
ACM
General Post Office
P.O. Box 30777
New York, NY 10087-0777

SIGAPP

Association for Computing Machinery

www.acm.org/joinsigs

Advancing Computing as a Science & Profession