

THE CONCURRENCY COLUMN

BY

MARINO MICULAN AND NOBUKO YOSHIDA

Department of Mathematics, Computer Science and Physics

University of Udine

Via delle Scienze 206, Udine, Italy

`marino.miculan@uniud.it`

Department of Computer Science

University of Oxford

Wolfson Building, Parks Road, Oxford, UK

`nobuko.yoshida@cs.ox.ac.uk`

Welcome to this installment of *The Concurrency Column*. In this issue, we are pleased to present a contribution by Luca Aceto, who collects his interviews with the recipients of the CONCUR Test-of-Time Award 2026. For the period 2006–2009, the Award Committee selected two papers: “A New Type System for Deadlock-Free Processes” by Naoki Kobayashi, presented at CONCUR 2006, and “Strategy Logic” by Krishnendu Chatterjee, Thomas A. Henzinger and Nir Piterman, presented at CONCUR 2007. From two rather different angles—type systems for the π -calculus and logics for games on graphs—both papers have durably shaped the way our community reasons about the behaviour of concurrent and interactive systems.

In the first interview, Naoki Kobayashi retraces the long lineage of type systems guaranteeing deadlock-freedom, from the early work on the linear π -calculus to the award-winning paper. The central design tension he describes is that between expressiveness and automation: “time tags” with their partial orders gave expressive power, capability and obligation levels gave automated type inference, and the CONCUR 2006 paper struck a balance between the two by combining a restricted form of the former with the latter. Kobayashi also discusses the role played by the TyPiCal implementation in validating and stress-testing the theory,

the encoding of the simply-typed λ -calculus with recursion as a yardstick for expressiveness, and the duality of obligations and capabilities that later resurfaced in the verification of security protocols and of lock-based concurrent programs. Looking ahead, he argues that the advent of GenAI-based coding agents may well overturn the design philosophy of “lightweight” formal methods: if annotations can be generated automatically, then precision and completeness may matter more than parsimony.

In the second interview, Krishnendu Chatterjee, Thomas Henzinger and Nir Piterman explain how Strategy Logic came about by treating strategies as explicit, first-class objects that the logic can quantify over. Their motivation was to express game-theoretic notions such as Nash and secure equilibria within a logical framework for games on graphs; the one-alternation fragment came first, and the full logic only afterwards. They discuss the interplay between linear- and branching-time reasoning that makes Strategy Logic and the alternating-time μ -calculus expressively incomparable, the tree-automata techniques underpinning their decidability and complexity results, the state of tool support, and the enthusiastic uptake of the logic by the multi-agent systems community. They also reflect on games on graphs as a genuinely unifying theme bridging the Volume A–Volume B divide, and on the open problems the paper left behind.

Beyond their technical content, these interviews are also research-life stories: they show how collaborations begin, how ideas mature over a decade, and how our field may reposition itself in an era in which AI-generated software needs formal checks more than ever. We hope you will enjoy reading them as much as we did.

INTERVIEWS WITH THE RECIPIENTS OF THE CONCUR TEST-OF-TIME AWARD 2026

Luca Aceto

ICE-TCS, Department of Computer Science,
Reykjavik University, Iceland
Gran Sasso Science Institute, L'Aquila, Italy
`luca@ru.is`, `luca.aceto@gssi.it`

The 2026 CONCUR Test-of-Time Award Committee (Anca Muscholl (chair), Javier Esparza and Prakash Panangaden) has selected the following two papers for that accolade for the period 2006–2009:

- Naoki Kobayashi, “A New Type System for Deadlock-Free Processes”, which appeared at CONCUR 2006, and
- Krishnendu Chatterjee, Thomas A. Henzinger and Nir Piterman, “Strategy Logic”, which appeared at CONCUR 2007.

I interviewed the awardees via email and posted their answers to my questions on my professional blog.¹

This piece collects those interviews essentially verbatim. In order to preserve the informal nature of the email discussions I had with Naoki, Krishnendu, Nir and Tom, I only made a small number of minor edits to correct a couple of typos and to add a bibliography.

I thank Naoki, Krishnendu, Nir and Tom for their willingness to answer my half-baked questions and for sharing their thoughts with the community in such an inspiring way. I hope you will enjoy reading their answers as much as I did.

1 Interview with Naoki Kobayashi

Luca: You receive the CONCUR Test-of-Time Award 2026 for your paper “A New Type System for Deadlock-Free Processes”, which appeared at CONCUR 2006 [9]. That article is the culmination of a series of contributions you gave on

¹<https://processalgebra.blogspot.com/>

the development of type systems for variations on the π -calculus that guarantee deadlock-freedom. Could you briefly explain to our readers how you came to study the question addressed in your award-winning article and how the main ideas in your CONCUR 2006 paper evolved over time? Which of the ideas and results in your paper did you find most pleasing, surprising or challenging?

Naoki: If I remember correctly, the idea of using types to ensure deadlock-freedom came up during discussions on the linear π -calculus with Benjamin Pierce and David Turner [10]. Indeed, if one wants to ensure that a channel is really used exactly once, one also has to ensure that the process does not get stuck before using it. An initial result on type systems for deadlock-freedom was published in LICS 1997 [7].

After that, the type system was extended in two directions: enabling automated type inference and increasing expressiveness. These were somewhat conflicting goals, and the CONCUR 2006 paper achieved a balance between them. The core idea was as follows. To ensure deadlock-freedom, it is necessary to control dependencies among different channels. In [7], such dependencies were expressed using “time tags” and a possibly infinite partial order on them. The time tags were later replaced in [8] by natural numbers, called capability and obligation levels, to enable automated inference [8], but at the cost of expressive power. The CONCUR 2006 paper combined a restricted form of time tags with capability and obligation levels, thereby recovering much of the expressive power while retaining automated inference.

What I found most pleasing was precisely this balance: the paper showed that one could make the type system significantly more practical without completely giving up the conceptual elegance and expressiveness of the earlier formulation.

Luca: The contribution in your article achieves a good trade-off between expressiveness of the type system and ease of implementing its associated type inference algorithm. Did you or any other researcher try to extend the range of examples that could be handled by a variation on your type system? Do you think that it would be worth doing so or have we hit the point of diminishing returns, where any advance would be very technical and hard to implement?

Naoki: With Elena Giachino and Cosimo Laneve, I indeed investigated such an extension in [4]. The system developed there can deal with infinite chains of dependencies more general than those handled in the CONCUR 2006 paper. It was, however, developed for value-passing CCS rather than the π -calculus. Incorporating the idea of [4] into the π -calculus setting remains future work. Another possible direction would be to combine the approach with generic types [6], which can capture precise dependencies among a set of channels. Unfortunately, I did not have time to fully explore this direction. As for whether it is worth pushing this line further, my feeling is that further general-purpose extensions may easily

become technically involved and hard to implement. Perhaps a more promising direction would be to adapt the type system to real concurrent languages such as Go, and then see what kinds of extensions are most beneficial in practice.

Luca: Commendably, you supported the theoretical developments in your article with an implementation in TyPiCal. What role did the implementation work play in your research on type systems guaranteeing deadlock-freedom? Did the theoretical developments benefit from the implementation? Do you think that every proposal for a type system for some language ought to be accompanied by an implementation?

Naoki: TyPiCal was initially developed as a proof of concept for the type inference algorithm of [8]. At that time, the theory had already been fully developed, and the implementation was mainly used to convince readers that the algorithm indeed works in practice. In later extensions of TyPiCal, for the CONCUR 2006 paper and [11], the implementation was also useful for checking the theories.

I think it is desirable for every type-system proposal to be accompanied by an implementation, although I would not say that it is absolutely necessary. An implementation is useful not only for checking the theory, but also for identifying limitations of the theory and suggesting possible directions for improvement.

Luca: You showed, amongst other things, that the simply-typed λ -calculus with recursion could be encoded in the deadlock-free fragment of your calculus. What role did this result play in convincing you that your type system was sufficiently expressive? What were the main challenges that you had to overcome in developing that encoding and what were the main inspirations for that result?

Naoki: The encodability of the simply-typed λ -calculus with recursion in the deadlock-free fragment was a sort of minimal requirement for expressiveness. Since that property had already been shown for the original deadlock-free type system [7], I wanted to show that a similar level of expressive power was retained in the CONCUR 2006 type system.

There was also a more conceptual reason for considering such an encoding. Deadlock-freedom corresponds to the progress property of well-typed functional programs: evaluation does not get stuck. Thus, the encodability result shows that this progress property can be preserved by compilation from possibly parallel functional programs into the π -calculus, which may be viewed as a CPS-style, lower-level language with concurrency primitives.

The main challenge was how to give a finite representation, within the type system, of the infinite chains of causal dependencies induced by recursion. As mentioned in the answer to the first question, we achieved this by combining the ideas of capability/obligation levels from [8] with a partial order on time tags from [7].

Luca: Your article gave a seminal contribution to the study of type systems for the π -calculus that guarantee some desirable properties. Other examples of such contributions are, for instance, your paper with Davide Sangiorgi on a type system for lock-freedom or the paper by Yuxin Deng and Davide Sangiorgi on ensuring termination by typability amongst others. In hindsight, are there any relations amongst those contributions or ideas that underlie them and that can guide further developments in a systematic fashion?

Naoki: All three type systems you mentioned are related in that they control causal dependencies using natural numbers called “levels”. The type system for deadlock-freedom forbids cyclic dependencies, while the type systems for lock-freedom and termination forbid infinite chains of dependencies. The formulation of such reasoning through types, however, suffers from inherent limitations in expressive power.

To address this limitation, my paper with Davide Sangiorgi [12] suggested a new direction: combining syntactic typing rules with semantic ones. The latter can be discharged by resorting to model checkers, interactive theorem provers, or other verification tools, thereby complementing the limitations of type systems. I think this combination of type-based reasoning and semantic verification may provide a useful guide for further developments: types can capture common structural patterns, while semantic methods can handle cases that fall outside the reach of purely syntactic typing rules.

Luca: Are there any problems that you left open in your award-winning article that you’d still love to see solved?

Naoki: I would still like to see at least two directions explored further. First, increasing the expressiveness of the type system by integrating the ideas of [4] and [6], mentioned above, would probably be necessary to make the approach practical. Second, since the type-based approach is inherently incomplete, finding a smooth integration with other methods, such as model checking and interactive theorem proving, remains an important open problem, as suggested in [11].

Luca: How did the results and the techniques you developed in your award-winning paper influence your subsequent research? Is there any result obtained by other researchers that builds on, or is related to, your work and that you like in particular?

Naoki: A key ingredient of our type systems for deadlock-freedom is the notion of obligations and capabilities. An obligation of a send or receive operation describes the need to perform that operation, while a capability describes a guarantee that the operation can succeed. For example, in a client-server model, a client has the capability to send a request successfully, while a server has an obligation to receive and handle the request. In lock-based mutual exclusion, a thread has the

capability to eventually acquire a lock, and, after acquiring the lock, it has an obligation to eventually release the lock.

These dual notions of obligations and capabilities turned out to be useful in other contexts. For example, we adapted them for automated verification of security protocols in [3]. Another line of work developed related ideas for more conventional concurrent programming languages. Leino et al. applied related techniques to deadlock-freedom of channels and locks in [13], and this line was further extended to monitors and condition variables by Hamin and Jacobs in [5]. I found this line of work particularly interesting because it shows that similar ideas arise naturally also in a more conventional programming-language setting.

Luca: What are the research topics related to type systems that you find most interesting right now? What role do you think type systems will or should have, if any, in the era of coding agents based on GenAI?

Naoki: I think GenAI may change not only type systems, but also the focus of formal methods more broadly. Traditionally, programs have been written by humans, and therefore “lightweight formal methods” have played an important role. In that setting, it was important to keep program annotations minimal and to automate verification or bug finding as much as possible, even if this sometimes came at the cost of precision or completeness.

In the era of GenAI-based coding agents, however, the situation may change. If coding agents can also generate annotations such as types and loop invariants, then the burden of writing annotations may no longer be such a serious issue. As a result, verification methods that aim at strong correctness guarantees, rather than merely finding bugs or checking lightweight properties, may become more important.

In this respect, the CONCUR 2006 type system can be seen as a product of the former design philosophy: it emphasized automation, at some cost in precision. A possible future direction would be to move beyond such lightweight type systems, by incorporating dependent types or combining type systems with other methods such as model checking, aiming at more complete verification methods.

Luca: What advice would you give to a young researcher who is keen to start working on topics related to types for programming languages?

Naoki: As discussed above, GenAI is rapidly changing the landscape of research in programming languages, and we are facing significant challenges in adapting to these changes and finding new research directions to explore. I think this is a very good opportunity for “AI-native” young researchers to play important roles. My advice would be to learn the classical foundations carefully, but at the same time not to be constrained by traditional assumptions about how programs are written and verified.

2 Interview with Krishnendu Chatterjee, Thomas Henzinger and Nir Piterman

Luca: You receive the CONCUR Test-of-Time Award 2026 for your paper “Strategy Logic”, which appeared at CONCUR 2007 [1] and, in archival form, in Information and Computation [2]. In that article, you introduced a seminal logic for expressing properties of strategies over two-player games on graphs. Could you briefly explain to our readers what the main features of strategy logic are? Could you also tell us how you came to study the question addressed in your award-winning article? Which of the results in your paper did you find most surprising or challenging?

Krishnendu, Nir and Tom (henceforth abbreviated to KNT): The defining feature of Strategy Logic is that it treats strategies as explicit, first-class objects: strategies are named by variables, and the logic can quantify over them. Our main motivation was to express central concepts from game theory, such as equilibria, within a logical framework for games on graphs. As for the results, what we find most appealing is that a logic this expressive still admits decidability, and that several natural fragments come with clean and reasonable computational complexity. Establishing these decidability and complexity results for various fragments was also the most technically challenging part of the work.

Luca: With the benefit of hindsight, having a logic to describe properties of games that treats strategies as first-class objects sounds like an extremely natural idea. However, previous logics such as ATL, ATL*, the alternating-time μ -calculus and game logic followed a different path. Do you recall how you came to the realisation that treating strategies explicitly was the “way to go”?

KNT: One of the key application areas for graph games has been reactive synthesis, and until 2004 reactive synthesis was studied primarily as an adversarial game. Hence logics such as ATL, ATL*, alternating-time μ -calculus, and game logic, all focus on strictly competitive or cooperative behaviors of game theory. Around 2004–2005, we started working on connections between algorithmic game theory and graph games. As a consequence we considered aspects of Nash equilibria and other not strictly competitive notions of game theory (such as secure equilibria) in graph games. A natural question was to build a logical framework that can express these aspects of game theory, which led to Strategy Logic. In fact we first isolated the one-alternation fragment, which suffices to express these equilibria, and only afterwards arrived at the full, more expressive logic. In hindsight, the shift to treating strategies explicitly was driven by the questions we were asking rather than by a single eureka moment.

Luca: Strategy logic builds on LTL, which is a very natural choice, IMHO. Did

you consider defining a version of strategy logic basing it on the (linear-time) modal μ -calculus? Would it be worth doing so and how would such a logic relate to the alternating-time μ -calculus?

KNT: There is always a tension between LTL and stronger formalisms that can recognize all ω -regular languages. Following the tradition of ATL and ATL*, we naturally chose to go with LTL for defining the (linear-time) objectives of players. As the techniques that we developed were automata based, it was clear that extensions of LTL that can express all ω -regular languages would be handled by the same techniques. The exact choice of the linear-time formalism (for example, ETL, QLTL, LDL, or the linear-time μ -calculus) is not very important as long as it can be readily translated to automata. In the alternating-time μ -calculus, however, by carefully nesting fixpoints and coalition quantification, we can define infinitely many changes of strategic context. But only in a completely adversarial manner.

Luca: As you are hinting, in your paper, amongst other results, you showed that the alternating-time μ -calculus and strategy logic have incomparable expressive power. Would a fixed-point version of strategy logic, based on the modal μ -calculus, be a way of achieving a logic that offers the “best of both worlds” while maintaining some of the good computational properties of strategy logic? Has anyone worked on such a logic?

KNT: We did not consider such a version of Strategy Logic. Intuitively, both classical and alternating-time μ -calculi combine local, single-transition branching operators such as Pre with fixpoint operators, whereas strategies express global behavioral choices and the objectives of players are defined on outcomes that are linear paths. An important expressive difference between the alternating-time μ -calculus and Strategy Logic is therefore due to the distinction between branching and linear time.

It seems interesting to consider a logic that would combine two types of pre operators: those that continue exploring pre-defined strategies and their induced behaviors and those that allow one to change the strategic context starting a new behavioral exploration. But we believe that the techniques that handle alternating-time μ -calculus would work for such a logic and the mix of behavior and control might be very hard to understand.

Luca: Did you or anyone else ever implement the model-checking algorithms you present in your award-winning paper? If the answer is negative, do you think that there would still be interest in such a model checker and in its experimental evaluation?

KNT: As far as we know there is no full implementation of Strategy Logic. In general, we have very good implementations supporting the manipulation and analysis of automata on infinite words (such as Spot and Owl). They are also

used as a basis for creating tools that solve reactive synthesis. But we do not have good tool support for using automata on infinite trees, which would be required in order to fully support Strategy Logic. The community studying Multi-Agent Systems adopted Strategy Logic and they have some support for the analysis of some questions. There are implementations of equilibrium checking and rational synthesis in the tool Eve that is developed in the group of Michael Wooldridge in Oxford. They support the analysis of concurrent game structures for such questions. There is also a restricted version of an epistemic extension of Strategy Logic that is included in the model checker MCMAS for Multi-Agent Systems, which was developed in the group of Alessio Lomuscio in Imperial College London.

MCMAS also supports ATL model checking and, in principle, it is possible to reduce the one-alternation fragment of Strategy Logic to ATL model checking, but we are not aware of this having been implemented.

Luca: You mentioned the uptake of strategy logic by the multi-agent systems community. How is strategy logic relevant and did you think that this work would be relevant to multi-agent systems?

KNT: Strategy Logic answered a natural need in Multi-Agent Systems research. For MAS, questions about the goals of agents and hence strategies are very natural. Many questions relate to rational behavior: whether agents have an incentive to follow a protocol, stability of behavior, what can coalitions do, and whether individuals can profitably deviate. The way Strategy Logic puts strategies in the center as explicit objects makes it very natural to study these questions.

We wouldn't say that we saw it coming, but the signs of early adoption of strategic reasoning by the MAS community were already there. The uptake of ATL and ATL* started in the early 2000s and by 2004–2005 people were using it regularly. We are also very happy that some of the major developments of Strategy Logic came from this community.

Luca: Are there any problems that you left open in your award-winning article that you'd still love to see solved? Did you or any colleagues study the problem of "strategy synthesis"?

KNT: Definitely. For the full logic, our paper established only a non-elementary upper bound and left the matching lower bound open. This gap was later closed by colleagues, who proved a matching non-elementary lower bound and thereby settled the computational complexity of the full logic. We also concentrated on the case of two-player games rather than multi-player games. The interaction between the logic and the game structure means that the two-player framework, in a sense, already captures the complexity of the logic. Indeed, the same techniques based on tree automata were later used by others to extend the logic to the multi-player setting (and concurrent game structures).

Luca: I am interested in how research collaborations start, as I like to tell “research-life stories” to PhD students and young researchers of all ages. Could you tell us how you started your collaboration on the award-winning paper?

KNT: The collaboration grew naturally out of a question from Tom (Thomas Henzinger): could notions such as the equilibria we had been studying be expressed in ATL or ATL*, and if not, what would be a natural and concise logic that could express them? Pursuing this led us to the one-alternation fragment. We then realized that strategies can be viewed as trees, which meant tree automata were the right tool—and Nir (Nir Piterman) was our automata expert. So the paper really came together at the meeting point of three ingredients: the study of non-zero-sum games, the wish for a logical framework to express their concepts, and tree-automata techniques.

Luca: How did the results and the techniques you developed in your award-winning paper influence your subsequent research? Is there any result obtained by other researchers that builds on your work and that you like in particular?

KNT: The result had a lasting influence. The interplay between games and automata that we exploited in the paper fed directly into later lines of work, e.g., from the connection of games and automata the notion of good-for-games (also known as history-deterministic) automata emerged. Among the results by others that build on Strategy Logic, the matching non-elementary lower bound and the work on special classes of strategies—such as the distinction between behavioral and non-behavioral strategies—are very elegant.

Luca: To my mind, games on graphs ought to be viewed as one of the unifying themes within TCS, bridging the Volume A-Volume B divide, and I am happy to see that there is a book-length treatment² covering the subject now. What is your view on this matter? What impact do you think that your work has had, if any, on the community working on algorithmic game theory, broadly construed? (I am reminded of the slides³ for a, typically thought-provoking, talk delivered by Moshe Vardi.) What has our community learnt from the work done in the field of computational game theory? And what, if anything, did they learn from the work done within the concurrency theory community?

KNT: Indeed, games on graphs is one of the unifying themes between Volume A and Volume B: it connects with logic and automata theory from Vol. B, and it connects with the notion of alternation from complexity theory and graph algorithms from Vol. A. The inflow of ideas from algorithmic game theory, or game theory in general, is a key source of ideas in this work, e.g., to express equilibria

²See <https://arxiv.org/abs/2305.10546>.

³<https://www.cs.rice.edu/~vardi/papers/sr15.pdf>

in a logical framework. Other notions of equilibria have also been studied in the context of reactive synthesis (e.g., rational synthesis), and Vol. B researchers have explored many concepts from general game theory beyond strictly competitive games. The broad area of algorithmic game theory also benefited from logical reasoning grounded in concurrency theory, e.g., Strategy Logic is the key logic for reasoning about behaviors in Multi-Agent Systems, and axiomatic characterizations for reasoning about strategy spaces have been influential in evolutionary game theory. There is also work using Strategy Logic in mechanism design.

Luca: What are the research topics related to logics for games on graphs that you find most interesting right now?

KNT: We can think of several interesting open questions: (i) the relationship between games and automata through the lens of history-determinism and similar concepts is an exciting direction; (ii) algorithmic bounds for several fundamental problems remain open, e.g., a polynomial-time algorithm for parity games; (iii) the degree to which sources of randomness can be shared between different players is an active area of research in concurrent games; and (iv) the study of computationally efficient logics which lie between ATL^* and Strategy Logic is also under-explored.

Luca: What advice would you give to a young researcher who is keen to start working on topics related to logics for games and other computational problems related to games?

KNT: The rapid progress in AI makes this question a difficult one. We believe that formal methods in general will become a central part of computing, more than they have ever been, because AI-generated software needs checks even more than software written by humans, but even more so, because modern AI can, for the first time, provide or at least support such checks on a scale that was hitherto impossible. Formal checks do not necessarily have to take the form of, say, Lean proofs, but they could also include state-based reasoning involving automata and games. It is always difficult to predict the future, but finding the right place for our field in this future seems a uniquely exciting opportunity.

References

- [1] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR 2007, 18th International Conference on Concurrency Theory*, volume 4703 of *Lecture Notes in Computer Science*, pages 59–73. Springer, 2007.
- [2] Krishnendu Chatterjee, Thomas A. Henzinger, and Nir Piterman. Strategy logic. *Inf. Comput.*, 208(6):677–693, 2010.

- [3] Morten Dahl, Naoki Kobayashi, Yunde Sun, and Hans Hüttel. Type-based automated verification of authenticity in asymmetric cryptographic protocols. In Tefik Bultan and Pao-Ann Hsiung, editors, *Automated Technology for Verification and Analysis, 9th International Symposium, ATVA 2011*, volume 6996 of *Lecture Notes in Computer Science*, pages 75–89. Springer, 2011.
- [4] Elena Giachino, Naoki Kobayashi, and Cosimo Laneve. Deadlock analysis of unbounded process networks. In Paolo Baldan and Daniele Gorla, editors, *CONCUR 2014, 25th International Conference on Concurrency Theory*, volume 8704 of *Lecture Notes in Computer Science*, pages 63–77. Springer, 2014.
- [5] Jafar Hamin and Bart Jacobs. Deadlock-free monitors. In Amal Ahmed, editor, *Programming Languages and Systems, 27th European Symposium on Programming, ESOP 2018*, volume 10801 of *Lecture Notes in Computer Science*, pages 415–441. Springer, 2018.
- [6] Atsushi Igarashi and Naoki Kobayashi. A generic type system for the Pi-calculus. *Theor. Comput. Sci.*, 311(1–3):121–163, 2004.
- [7] Naoki Kobayashi. A partially deadlock-free typed process calculus. In *LICS 1997, 12th Annual IEEE Symposium on Logic in Computer Science*, pages 128–139. IEEE Computer Society, 1997.
- [8] Naoki Kobayashi. Type-based information flow analysis for the pi-calculus. *Acta Informatica*, 42(4–5):291–347, 2005.
- [9] Naoki Kobayashi. A new type system for deadlock-free processes. In Christel Baier and Holger Hermanns, editors, *CONCUR 2006, 17th International Conference on Concurrency Theory*, volume 4137 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2006.
- [10] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the pi-calculus. In Hans-Juergen Boehm and Guy L. Steele Jr., editors, *Conference Record of POPL’96: The 23rd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 358–371. ACM Press, 1996.
- [11] Naoki Kobayashi and Davide Sangiorgi. A hybrid type system for lock-freedom of mobile processes. In Aarti Gupta and Sharad Malik, editors, *Computer Aided Verification, 20th International Conference, CAV 2008*, volume 5123 of *Lecture Notes in Computer Science*, pages 80–93. Springer, 2008.
- [12] Naoki Kobayashi and Davide Sangiorgi. A hybrid type system for lock-freedom of mobile processes. *ACM Trans. Program. Lang. Syst.*, 32(5):16:1–16:49, 2010.
- [13] K. Rustan M. Leino, Peter Müller, and Jan Smans. Deadlock-free channels and locks. In Andrew D. Gordon, editor, *Programming Languages and Systems, 19th European Symposium on Programming, ESOP 2010*, volume 6012 of *Lecture Notes in Computer Science*, pages 407–426. Springer, 2010.