

On the Computational Complexity of Bisimulation, Redux

Faron Moller
Department of Computer Science
University of Wales Swansea
Singleton Park, Swansea SA2 8PP, UK
F.G.Moller@swansea.ac.uk

Scott A. Smolka
Department of Computer Science
SUNY at Stony Brook
Stony Brook, NY 11790-4400, USA
sas@cs.sunysb.edu

Categories and Subject Descriptors

F.2.2 [Nonnumerical Algorithms and Problems]: Complexity of proof procedures; F.1.1 [Models of Computation]: Automata (e.g., finite, push-down, resource-bounded); D.2.4 [Software/Program Verification]: Formal methods

General Terms

Theory, Verification

1. PREFACE

This is a revised and updated version of [28]. Paris Kanellakis and the second author (Smolka) were among the first to investigate the computational complexity of bisimulation, and the first author (Moller) has a long-established track record in the field. The authors therefore believe that the proceedings of *Principles of Computing and Knowledge: Paris C. Kanellakis Memorial Workshop* represent an ideal opportunity for another look at the subject.

2. INTRODUCTION

In his Turing Award lecture [12], Juris Hartmanis eloquently discusses, among other things, the fundamental rôle that *computational complexity* theory plays in computer science. He goes on, in the context of describing joint work with Phil Lewis and Richard Stearns, to highlight some of the results obtained on the computational complexity of problems in formal language theory; e.g., all context-free languages are contained in $\text{TIME}[n^3]$ and $\text{SPACE}[\log^2 n]$.

We argue here that the computational complexity of *generative devices* such as grammars or automata takes on a new and interesting light when such devices are interpreted as generating (concurrent) *processes* rather than formal languages, and the traditional notion of language equivalence is replaced by *bisimulation* equivalence. A grammar in Greibach Normal Form (GNF) describes a process represented as a state transition system in the following fashion: the states

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

PCK50, June 8, 2003, San Diego, California, USA.

This is a revision of the work published in *ACM Computing Surveys*, 27(2): 287-289(1995), (<http://doi.acm.org/10.1145/210376.210406>).

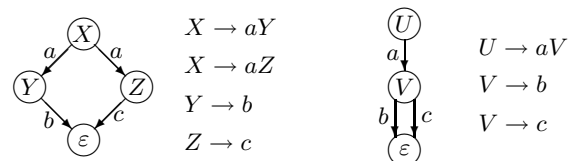
Copyright 2003 ACM 1-58113-604-8/03/0006 ...\$5.00.

of the process correspond to sequences of nonterminals; and the transitions leading from a state corresponding to a sequence starting with a nonterminal X are prescribed, in a one-to-one fashion, by the rules of the grammar corresponding to the nonterminal X . Formally, $X\beta \xrightarrow{a} \alpha\beta$ if $X \rightarrow a\alpha$ is a rule of the grammar. Thus, each nonterminal itself represents a (state in a) process, and the leftmost derivation rule reflects the idea that the concatenation of nonterminals models the sequential composition of the processes represented by the nonterminals.

Bisimulation is the cornerstone of a number of theories of concurrent and distributed computing, most notably Robin Milner's Calculus of Communicating Systems (CCS) [24] and the π -calculus [26]. Given that Milner received the 1991 Turing Award and that bisimulation figured prominently in his Turing Award lecture [25], we believe it is particularly apropos to re-examine the field of computational complexity—whose inception is due to Hartmanis and Stearns—in this setting.

Bisimulation has also drawn the attention of modal logicians, who called it the “zig-zag” relation some 20 years ago. It is intimately related to the distinguishing powers of general branching-time temporal logics, in particular the modal μ -calculus. Set theorists have been attracted to bisimulation, as it forms the basis of Peter Aczel's Anti Foundation Axiom for non-well-founded set theory [2]. The functional programming community has also shown interest in bisimulation, as evidenced by Samson Abramsky's notion of *applicative bisimulation* for relating terms of the lazy lambda calculus [1]. Finally, as we shall demonstrate, bisimulation has an elegant game-theoretic interpretation as promulgated by Colin Stirling [39].

To motivate this study, consider the regular expressions $ab + ac$ and $a(b + c)$ which are represented by the following nondeterministic finite-state automata (NFA) and corresponding regular grammars.



These expressions, as well as their corresponding automata, are clearly language equivalent, as they both describe the language $\{ab, ac\}$; as language generators, they are indistinguishable. However, viewed as process generators they may be distinguished. The first automaton in its initial state X may perform an a -transition and evolve into either state

Y —from which only a b -transition is possible—or state Z —from which only a c -transition is possible; on the other hand, the second automaton in its initial state U performs the a -transition and evolves into state V from which both the b - and c -transitions are possible.

If we interpret these automata as representing the behaviours of processes, with the transitions being potential communications with the environment in which these processes reside, then it behooves us to consider them as behaviourally *inequivalent*. For example, after the initial communication involving the a -transition, the second process would be in state V from which it is willing to participate in a communication involving a b -transition, whereas the first process may be in state Z from which it will refuse to participate in a communication involving a b -transition. In the terminology of concurrency theory, the first process may *deadlock* in an instance in which the second will not.

Milner [23] proposed *bisimilarity* to formally capture the notion of behavioural equivalence, and gave it, along with David Park [32], a simple and elegant mathematical definition in terms of bisimulations. A *bisimulation* is a binary relation \mathcal{R} on processes such that whenever $\mathcal{R}(P, Q)$, if P can perform some a -transition to become P' then Q can perform the same a -transition to become some Q' such that $\mathcal{R}(P', Q')$; and conversely if Q can perform some a -transition to become Q' then P can perform the same a -transition to become some P' such that $\mathcal{R}(P', Q')$. Note the recursive nature of the definition. Now, two processes P and Q are bisimilar if there exists a bisimulation \mathcal{R} such that $\mathcal{R}(P, Q)$. It is well-known that bisimulations are closed under union and that the largest bisimulation, under set inclusion, exists. In fact, this largest bisimulation, \sim , is an equivalence relation and taken to be bisimulation equivalence.

We now have in place the three main ingredients of a formal language theory in a new setting: automata and grammars (processes), and equivalence (bisimilarity). The computational complexity of bisimulation in this formal-language framework, however, differs greatly from its classical counterpart, with a number of surprising twists and turns worth mentioning. We concentrate here on the inner layers of the Chomsky hierarchy, viz. *regular* and *context-free* processes, and note in passing that a language like CCS is easily shown to be Turing-powerful.

3. REGULAR PROCESSES

In the case of regular processes, that is, those given by right-linear GNF grammars such as the two depicted above, the main complexity result is as follows. Let P and Q be regular processes whose underlying NFA have a total of n states and m transitions. Then, as was shown by Kanelakis and Smolka [20], whether or not P and Q are bisimilar can be decided in polynomial time, $O(nm)$ time to be exact. (This algorithm was subsequently improved upon by Paige and Tarjan who devised one that runs in $O(m \log n)$ time [30].) This is in stark contrast to the equivalence problem for regular expressions, which was shown by Stockmeyer and Meyer to be PSPACE-complete [43].

Moreover, bisimulation was originally defined by Milner as the limit of a sequence of successively finer equivalence relations \sim_k , where \sim_1 is language equivalence. Kanelakis and Smolka showed that, for each fixed k , deciding \sim_k is PSPACE-complete, a complexity that disappears in

the limit; i.e., upon reaching \sim .

4. CONTEXT-FREE PROCESSES

The situation is even more dramatic in the context-free case, where the resulting processes are no longer finite-state. In the concurrency theory community, context-free processes are referred to as BPA (Basic Process Algebra) processes. In the classical setting, Bar-Hillel, Perles, and Shamir [5] showed that the equivalence problem for languages generated by nondeterministic context-free grammars is undecidable. Taking advantage of the periodic structure exhibited by bisimilar processes, Baeten, Bergstra, and Klop [3] were able to show that bisimilarity of *normed* BPA—those context-free processes in which the underlying GNF grammar contains no redundant nonterminals—is decidable. (Being normed means that there is a sequence of transitions leading from every state of the process to the final state ε . The norm of a state is defined to be the length of a shortest such sequence.) In fact, Hirshfeld, Jerrum and Moller [14] showed that, in this case, bisimilarity can be decided in polynomial time. Restricting to simple (i.e., deterministic) normed grammars, where language equivalence and bisimilarity coincide, this gives that language equivalence is polynomially decidable, improving vastly on the doubly-exponential algorithm of Korenjak and Hopcroft [21].

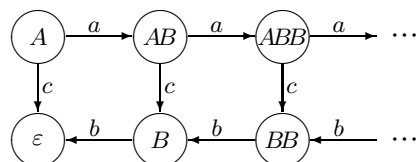
For arbitrary (unnormed) BPA processes, Christensen, Hüttel, and Stirling [11] showed that bisimilarity is still decidable. However, the complexity in this general case is now known to be PSPACE-hard [37], yet no worse than doubly-exponential [8].

5. COMMUTATIVE CONTEXT-FREE PROCESSES

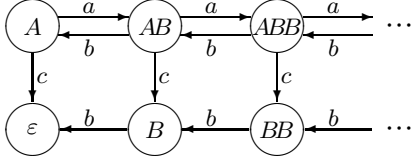
Of course, in studying concurrent processes one would like to consider processes composed not merely sequentially as with context-free processes, but concurrently as well. A simple form of concurrent composition can be modelled by considering *commutative* context-free processes, that is, where we now interpret concatenation of nonterminals modulo commutativity. In this way, the leftmost derivation rule allows any nonterminal in a sequence to be used to provide the next transition from the state associated with that sequence, simply by commuting the sequence to bring the relevant nonterminal to the start of the sequence. Concatenation of nonterminals in this case models the parallel composition of the processes represented by the nonterminals, rather than the sequential composition as in the case of the earlier context-free processes. In the concurrency theory community, the resulting process is referred to as a BPP (Basic Parallel Process) process. For example, the grammar

$$\begin{aligned} A &\rightarrow aAB \\ A &\rightarrow c \\ B &\rightarrow b \end{aligned}$$

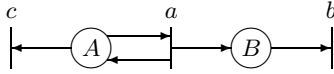
gives rise to the BPA (context-free) process



and to the BPP (commutative context-free) process



BPP processes correspond to communication-free Petri nets, those (place/transition) Petri nets in which each transition has a unique input place. For example, the above BPP process corresponds to the following Petri net:



The results regarding deciding bisimulation in this case are similar to those for BPA processes. Hirshfeld [13] showed that, once again, language equivalence is undecidable, while Christensen, Hirshfeld and Moller [9] showed that bisimilarity is decidable in general, and Hirshfeld, Jerrum and Moller [15] showed that it is decidable in polynomial time. Jančar [16] has recently demonstrated that the general problem is PSPACE-complete.

One noteworthy corollary from the latter paper is the resolution of an intriguing long-standing conjecture. In the case of BPA processes, if X is an unnormed variable it is easily confirmed that $X \sim X\alpha$ for any sequence α ; being unnormed, the nonterminal X represents a process which can never terminate, so the behaviour of $X\alpha$ will be the same as that of X . Also, if $XX \sim XXX$ then the variable X must be unnormed; this follows from the fact that the norm is additive ($norm(\alpha\beta) = norm(\alpha) + norm(\beta)$), and that bisimilar processes must have the same norm. Thus it is clear that the identity $X \sim XX$ follows immediately from $XX \sim XXX$. However, this is by no means obvious in the case of BPP processes. This conjecture was put forward ten years ago (a stronger version appears in [10]), and since then many clever researchers have failed at every attempt to prove this cancellation law; none of the standard bisimulation proof techniques seem to be applicable to this question. Jančar [16] finally provided a proof which is unexpectedly complicated for such a simple-looking conjecture.

6. STATE-EXTENDED PROCESSES

A common extension to context-free and commutative context-free processes is provided by including a finite-state control. With such an extension, the grammar rules are no longer based solely on the leading nonterminal of the sequence representing the state, but are dictated as well by the finite-state control. State-extended BPA naturally correspond to pushdown automata (with the sequence of nonterminals representing the contents of the stack), while state-extended BPP correspond to multiset automata, and represent a subclass of Petri nets.

Sénizergues [34] and Stirling [41] both showed the decidability of bisimulation equivalence over state-extended BPA. A more noteworthy related result by Stirling [42] is that bisimilarity is decidable over strict deterministic grammars. This result reinforces (and gives a shorter proof for) Sénizergues' solution [35] to the long-standing problem of language equivalence between deterministic pushdown automata.

For the case of state-extended BPP, the result differs from the sequential case; in this case, bisimilarity is undecidable [27].

7. BISIMULATION AND OTHER GAMES

It is instructive to view bisimulation equivalence in terms of particular two-player games. A *game* is provided by a pair of processes (P, Q) , with the players alternating moves as follows: Player 1 chooses a transition of one of the processes, and in response, Player 2 must choose an identical transition of the other process. The game then continues starting from the resulting pair of processes. If Player 2 ever finds that she cannot respond to a move made by Player 1, she loses the game. A moment's reflection then leads to the realization that Player 2 has a defending strategy exactly when the two processes are bisimilar: if there is a bisimulation relating the processes, then a defending strategy for Player 2 consists of merely matching transitions made by Player 1 which lead to a resulting pair which is also contained in the bisimulation relation. Conversely Player 1 has a winning strategy exactly when the two processes are not bisimilar.

As an example, processes X and U pictured above are not bisimilar. (Try constructing a bisimulation relating X and U ; there isn't any!) This nonbisimilarity is evidenced by the existence of an obvious winning strategy for Player 1 in the game defined by the pair (X, U) . After one exchange of moves consisting of a single a -transition, the game must be in either the configuration (Y, V) or (Z, V) . In the first instance, Player 1 may win by choosing the single c -transition from process V , while in the second instance she may win by choosing the single b -transition from process V . Thus, bisimilarity is a strictly more discriminating equivalence relation than language equivalence, and is intrinsically sensitive (unlike language equivalence) to the nondeterministic branching structure of processes.

The simple game described above for characterising bisimulation can be adapted to other equivalences, such as those reflecting a noninterleaving semantic interpretation¹ [29]. More interestingly, they can be modified to reflect model-checking algorithms for established modal and temporal logics [38, 40]. For example, rules may be introduced allowing players to mark selected subsets of states, and to permit them to jump to arbitrary marked states. Such rules are typical when characterising fixed points or second-order quantification in logics. In this way we get intuitive explanations of when a process satisfies a specification presented as a logical property: the question (and its complexity) is reduced to the question (and the complexity) of determining the holder of the winning strategy in a game.

Many classes of games, and their decision complexity, have been extensively studied over the last years. Yet many fundamental and practically important problems remain open. In particular, we mention the following two classes of games

¹In a setting of multiple, concurrently executing processes, bisimulation equivalence is reflective of an *interleaving* semantic interpretation of concurrency. Under this interpretation, the concurrent execution of actions a and b is captured by the regular expression $ab+ba$, reflecting the two possible interleavings of these actions. In a *noninterleaved* interpretation, a and b are thought to occur in a truly concurrent manner. Mathematically, this amounts to introducing some sort of *concurrency relation* indicating when two actions occur concurrently.

which are especially relevant for computer-aided verification.

Büchi Games. Given a finite directed bipartite graph $G = (V_1, V_2, E)$ and some $F \subseteq V_1$, can Player 1 (starting in $v \in V_1$ and selecting outgoing edges from V_1) force Player 2 (selecting outgoing edges from V_2) to visit vertices in F infinitely often?

Different standard fairness properties of real-life reactive systems are naturally formulated in terms of Büchi games, and the precise computational complexity of deciding the winner in such games is of primary practical concern. The original algorithm due to Rabin discovered three decades ago has quadratic time complexity [44] and has yet to be improved. For practical applicability it is crucial to find an algorithm with a subquadratic upper bound.

Parity Games. Parity games differ from Büchi games only in the winning condition. Given a sequence $F = (F_0, \dots, F_k)$ of disjoint subsets of V_1 , can Player 1 assure that no matter how cleverly Player 2 moves, in the resulting sequence some vertex from some F_{2i} occurs infinitely often, while all vertices from F_{2i+j} ($j > 0$) only finitely often?

Parity games are known to be polynomial-time equivalent to the fundamental μ -calculus model checking problem: whether a given Kripke model satisfies a propositional μ -calculus formula involving boolean connectives, the modal “next”, and least and greatest fixed-point operators. The μ -calculus is a powerful logic subsuming all other temporal logics of programs. Therefore, it is important to know precisely the complexity of its model checking. Currently, it is only known that the problem belongs to the class $NP \cap co-NP$; the best decision procedures are exponential. Any progress towards polynomial-time decidability of the problem will constitute a major theoretical contribution with important practical applications.

Finally, we would like to comment on more general games and Nash Equilibria. A *game*, in general, consists of n players, with Player i choosing a *strategy* $\sigma_i \in S_i$, along with functions $w_i : S_1 \times \dots \times S_n \rightarrow \mathbb{R}$ ($1 \leq i \leq n$) assigning a payoff to each person. *Rational behaviour* is defined as a *Nash equilibrium*: a combination $(\sigma_1, \dots, \sigma_n) \in S_1 \times \dots \times S_n$ of strategies for which for all i and all $\sigma'_i \in S_i$: $w_i(\sigma_1, \dots, \sigma_i, \dots, \sigma_n) \geq w_i(\sigma_1, \dots, \sigma'_i, \dots, \sigma_n)$; that is, no player has an incentive to deviate.

A Nash equilibrium need not exist, nor be unique. However, a corollary of the ingenious results by Nash himself is that a *mixed* (randomized) Nash equilibrium is always guaranteed to exist. In a game between *two* players each offered a *finite* set of strategies, the problem of computing a mixed Nash equilibrium is in NP but unlikely to be NP-hard. The importance of using Nash equilibria to understand the behaviour of distributed systems, particularly the Internet, is being exposed by various researchers, such as Koutsoupias and Papadimitriou [22]. Indeed, according to Papadimitriou [31]:

“Together with factoring, the complexity of finding a Nash equilibrium is in my opinion the most important concrete open question on the boundary of P today.”

8. PARALLEL COMPLEXITY

An intriguing question to ask about bisimulation is does it have an efficient *parallel* solution? The class NC contains those problems that can be solved in polylogarithmic time

using a polynomial number of processors (in the size of the input). NC is generally regarded as the class of problems that have fast parallel solutions.

It is generally believed that P-complete problems cannot be in NC. A problem is in P if it can be solved by a deterministic Turing machine in polynomial time. A problem is P-complete if it is in the class P, and it is P-hard in the sense that any other problem in P is log-space reducible to it. A reduction is log-space if it uses at most a logarithmic amount of intermediate storage space.

Balcazar et al. [4] established the P-completeness of the bisimulation checking problem for regular processes via a log-space reduction from the *Monotone Alternating Circuit Value Problem*. Despite this negative result, several parallel and distributed algorithms for deciding bisimulation equivalence of finite-state processes have been proposed that achieve non-trivial speedups in practice [45, 19, 33, 6].

9. CONCLUSIONS

We have offered a brief history of the computational complexity of bisimulation. Several comprehensive surveys about the subject, focusing on *infinite-state processes*, have been written (eg, [27, 18, 17]), including a major Handbook chapter [7]. There is even now a project devoted to maintaining an up-to-date overview of the state-of-the-art in this dynamic research topic [36].

The reader may have noticed the following trend about bisimulation: bisimulation equivalence is computationally easier to decide than language equivalence, regardless of the nature of the underlying process model, be it finite-state or infinite-state. It is interesting to search for an explanation to this computational dichotomy. Some insight can be gained by again noting that bisimulation is a much more discriminating equivalence than language equivalence, to the point where it is easier to decide. In particular, bisimilar states, for any symbol a , must lead to bisimilar states. The absence of this restriction on language-equivalent states in some sense forces one to determinize the automata in question to decide equivalence, a costly proposition indeed.

Finally, we ask what are the *practical ramifications* of the computational dichotomy? Happily, the answer is a positive one for computer scientists interested in bisimilarity, such as concurrency theorists and verification tool builders. In this case, one is confronted with a computationally tractable problem even for processes of a highly expressive nature.

10. REFERENCES

- [1] S. Abramsky. The lazy lambda calculus. In D. A. Turner, editor, *Research Topics in Functional Programming*, pages 65–116. Addison-Welsey, Reading, MA, 1990.
- [2] P. Aczel. *Non-Well-Founded Sets*. Number 14 in CSLI Lecture Notes. CSLI Publications, Stanford, CA, 1988.
- [3] J.C.M. Baeten, J.A. Bergstra, and J.W. Klop. Decidability of bisimulation equivalence for processes generating context-free languages. *Journal of the ACM*, 40(3):653–682, July 1993.
- [4] J. Balcazar, J. Gabarro, and M. Santha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4:638–648, 1992.
- [5] Y. Bar-Hillel, M. Perles, and E. Shamir. On formal properties of simple phrase structure grammars.

Zeitschrift für Phonetik, Sprachwissenschaft, und Kommunikationsforschung, 14:143–177, 1961.

- [6] S. Blom and S. Orzan. A distributed algorithm for strong bisimulation reduction of state spaces. In L. Brim and O. Grumberg, editors, *Parallel and Distributed Model Checking (PDMC 2002)*. Elsevier Science, Electronic Notes in Theoretical Computer Science, 68(4), 16 pages, 2002.
- [7] O. Burkart, D. Caucal, F. Moller, and B. Steffen. Verification on infinite structures. In J.A. Bergstra, A. Ponse, and S.A. Smolka, editors, *Handbook of Process Algebra*, chapter 9, pages 545–623. Elsevier Science, 2001.
- [8] O. Burkart, D. Caucal, and B. Steffen. An elementary decision procedure for arbitrary context-free processes. In *Proceedings of the 20th International Symposium on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, pages 423–433. Springer-Verlag, 1995.
- [9] S. Christensen, Y. Hirshfeld, and F. Moller. Bisimulation is decidable for Basic Parallel Processes. In *Proceedings of CONCUR'93: Concurrency Theory*, volume 715 of *Lecture Notes in Computer Science*, pages 143–157. Springer-Verlag, 1993.
- [10] S. Christensen, Y. Hirshfeld, and F. Moller. Decomposability, decidability and axiomatisability for bisimulation equivalence on Basic Parallel Processes. In *8th Annual IEEE Symposium on Logic in Computer Science*, pages 386–396. IEEE, 1993.
- [11] S. Christensen, H. Hüttel, and C. Stirling. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Information and Computation*, 12(2):143–148, 1995.
- [12] J. Hartmanis. Turing Award lecture: On computational complexity and the nature of computer science. *Communications of the ACM*, 37(10):37–43, October 1994.
- [13] Y. Hirshfeld. Petri nets and the equivalence problem. In *Proceedings of CSL'93: Computer Science Logic*, volume 832 of *Lecture Notes in Computer Science*, pages 165–174. Springer-Verlag, 1994.
- [14] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimilarity of normed context-free processes. *Theoretical Computer Science*, 158:143–159, May 1996.
- [15] Y. Hirshfeld, M. Jerrum, and F. Moller. A polynomial algorithm for deciding bisimulation equivalence of normed Basic Parallel Processes. *Mathematical Structures in Computer Science*, 6:251–259, 1996.
- [16] P. Jančar. Strong bisimilarity on Basic Parallel Processes is PSPACE-complete. In *18th Annual IEEE Symposium on Logic in Computer Science*. IEEE, 2003.
- [17] P. Jančar and A. Kučera. Equivalence-checking with infinite-state systems: Techniques and results. In *Proceedings of 29th Annual Conference on Current Trends in Theory and Practice of Informatics (SOFSEM'02)*, volume 2540 of *Lecture Notes in Computer Science*, pages 41–73. Springer-Verlag, 2002.
- [18] P. Jančar and F. Moller. Techniques for decidability and undecidability of bisimilarity. In *Proceedings of CONCUR'99: Concurrency Theory*, volume 1664 of *Lecture Notes in Computer Science*, pages 30–45. Springer-Verlag, 1999.
- [19] C. Jeong, Y. Kim, H. Kim, and Y. Oh. A faster parallel implementation of Kanellakis-Smolka algorithm for bisimilarity checking. In *Proceedings of the International Computer Symposium*, Tainan, Taiwan, 1998.
- [20] P. C. Kanellakis and S. A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, May 1990.
- [21] A.J. Korenjak and J.E. Hopcroft. Simple deterministic languages. In *7th Annual IEEE Symposium on Switching and Automata Theory*, pages 36–46. IEEE, 1966.
- [22] E. Koutsoupias and C. H. Papadimitriou. Worst-case equilibria. In *Proceedings of 16th Annual Symposium on Theoretical Aspects of Computer Science*, volume 1563 of *Lecture Notes in Computer Science*, pages 404–413. Springer-Verlag, 1999.
- [23] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *LNCS*. Springer-Verlag, Berlin, 1980.
- [24] R. Milner. *Communication and Concurrency*. International Series in Computer Science. Prentice Hall, 1989.
- [25] R. Milner. Elements of interaction — Turing Award lecture. *Communications of the ACM*, 36(1):78–89, January 1993.
- [26] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Parts I and II. *Information and Computation*, 100, 1992.
- [27] F. Moller. Infinite results. In *Proceedings of CONCUR'96: Concurrency Theory*, volume 1119 of *Lecture Notes in Computer Science*, pages 195–216. Springer-Verlag, 1996.
- [28] F. Moller and S. A. Smolka. On the computational complexity of bisimulation. *ACM Computing Surveys*, 27(2):287–289, June 1995.
- [29] M. Nielsen and C. Clausen. Bisimulation, games and logic. In *Results and Trends in Theoretical Computer Science: Colloquium in Honor of Arto Salomaa*, volume 812 of *Lecture Notes in Computer Science*, pages 289–306. Springer-Verlag, 1994.
- [30] R. Paige and R. E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, December 1987.
- [31] C. H. Papadimitriou. Algorithms, games, and the internet. In *Proceedings of 33rd Symposium on Theory of Computing*, pages 749–753. ACM Press, New York, 2001.
- [32] D. M. R. Park. Concurrency and automata on infinite sequences. In *Proceedings of 5th G.I. Conference on Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.
- [33] S. Rajasekaran and I. Lee. Parallel algorithms for relational coarsest partition problems. *IEEE Transactions on Parallel and Distributed Systems*, 9(7):687–699, July 1998.
- [34] G. Senizergues. Decidability of bisimulation equivalence for equational graphs of finite out-degree.

- In *39th Annual IEEE Symposium on Foundations of Computer Science*, pages 120–129. IEEE, 1998.
- [35] G. Senizergues. $L(A)=L(B)$? Decidability results from complete formal systems. *Theoretical Computer Science*, 251(1-2):1–166, January 2001.
- [36] J. Srba. Roadmap of infinite results. Department of Computer Science, University of Aarhus, <http://www.brics.dk/~srba/roadmap>, 2002.
- [37] J. Srba. Strong bisimilarity and regularity of Basic Process Algebra is PSPACE-hard. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming (ICALP'02)*, volume 2380 of *Lecture Notes in Computer Science*, pages 716–727. Springer-Verlag, 2002.
- [38] C. Stirling. Local model checking games. In *Proceedings of 6th International Conference on Concurrency Theory (CONCUR'95)*, volume 962 of *Lecture Notes in Computer Science*, pages 1–11. Springer-Verlag, 1995.
- [39] C. Stirling. Games for bisimulation and model checking. In *Notes for Mathfit Workshop on Finite Model Theory*, University of Wales, Swansea, July 1996.
- [40] C. Stirling. Modal and temporal logics for processes. In *Logics for Concurrency: Structure versus Automata*, volume 1043 of *Lecture Notes in Computer Science*, pages 149–237. Springer-Verlag, 1996.
- [41] C. Stirling. Decidability of bisimulation equivalence for pushdown processes. Research Report EDI-INF-RR-0005, School of Informatics, Edinburgh University, January 2000.
- [42] C. Stirling. Decidability of DPDA equivalence. *Theoretical Computer Science*, 255(1-2):1–31, March 2001.
- [43] L. J. Stockmeyer and A. Meyer. Word problems requiring exponential time. In *Proceedings of Fifth ACM Symposium on Theory of Computing*, pages 1–9, 1973.
- [44] M. Y. Vardi and P. Wolper. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Science*, 32(2):182–221, 1986.
- [45] S. Zhang and S. A. Smolka. Efficient parallelization of equivalence checking algorithms. In M. Diaz and R. Groz, editors, *Proceedings of Fifth International Conference on Formal Description Techniques*, pages 133–146, October 1992.