Int J Softw Tools Technol Transfer
DOI 10.1007/s10009-016-0424-3

INTRODUCTION

Parameterized verification

Parosh A. Abdulla[1] · **Giorgio Delzanno**[2]

Abstract The goal of parameterized verification is to prove the correctness of a system specification regardless of the number of its components. The problem is of interest in several different areas: verification of hardware design, multithreaded programs, distributed systems, and communication protocols. The problem is undecidable in general. Solutions for restricted classes of systems and properties have been studied in areas like theorem proving, model checking, automata and logic, process algebra, and constraint solving. In this introduction to the special issue, dedicated to a selection of works from the Parameterized Verification workshop PV '14 and PV '15, we survey some of the works developed in this research area.

Keywords Formal verification · Program analysis · Concurrent and distributed systems

Problem statement To ensure the quality of applications running on modern computer systems, we often face the problem of validating subsystems consisting of several instances of replicated components. This kind of scenario occurs at all possible level of abstractions of a computer system:

- In cloud architectures, sensor networks, and systems based on the Internet of Things arbitrarily large sets of heterogeneous components (client applications, sensors, and devices) are connected via multiple sessions to service providers.
- In communication protocols, distributed and multi-agent systems, business protocols, and access control policies used to control data privacy, the parameter to consider is the number of principals in combination with multiple sessions, structured data, time-stamps, etc.
- Algorithms for controlling race conditions, e.g., in multicore systems, are designed to work for an arbitrary number of system processes or software threads. The same property usually holds for concurrent data structures, design patterns for concurrency, and thread-safe software libraries. In this setting, there are several levels to consider ranging from lock-free algorithms to high level libraries or primitives like those provided by languages like Java, C#, and Erlang.
- At lower abstraction levels, we often encounter modular specifications of hardware designs. For instance, consistency and bus protocols are designed to work for families of circuits. Compositional verification of hardware design was one of the inspiring problem for parameterized verification [21,46,57].

Concrete examples of systems that are often formulated for an arbitrary number of components are:

- generalized versions of mutual exclusion protocols for multithreaded programs [27,59],
- mutual exclusion protocols with weak memory models [6],
- distributed algorithms for arbitrary network configurations [37,38],
- specifications of cache coherence protocols [29,40,46,57],
- cryptographic protocols with unbounded principals and sessions [1,65],
- multithreaded programs [50,51].

✉ Parosh A. Abdulla
parosh@it.uu.se

1 Uppsala University, Uppsala, Sweden

2 DIBRIS, Università di Genova, Genova, Italy

Published online: 24 June 2016

Verification problems for instances of all the above-mentioned systems are often taken as benchmarks for automated methods and tools developed in the field.

A general definition A common formalism used in formal system specifications is that of transition systems. A transition system is usually defined by a set of configurations C and by a relation $\rightarrow$ used to represent successor (and predecessor) states. An execution is then a sequence of states $s_1, \ldots, s_n, \ldots$ in which $s_i \rightarrow s_{i+1}$ for $i \geq 0$. All possible executions starting from a given configuration can be viewed as the game arena in which to play the verification game. Parameterized systems can be viewed then as transition systems $T(n)$ parametric in the number n of components. Parameterized verification amounts to checking a given property, function of n, $\varphi(n)$, for any possible value of n. Properties can be formulated in specification languages like temporal logic, a conservative extension of propositional logic with modal operators used to navigate along states and branches of the state space of a given transition system.

Parameterized verification is aimed at developing (semi) automated verification methods for solving verification problems for families of transition systems see, e.g., [21,27,36–40,46,57,59]. There are several parameters to be considered in this setting:

- the number of components,
- the behavior of each component (finite-state or infinite-state),
- the type of data manipulated in the system (finite vs infinite domains, structured data, recursive data structures, higher order data),
- the communication topology that interconnects components,
- the presence of time-sensitive operations,
- the type of communication primitives (synchronous vs asynchronous communication).

Every single parameter poses interesting questions for automated verification. Indeed, contrary to finite-state verification, parameterized verification is an undecidable problem in general [16,17]. Undecidability holds even for restricted properties like safety and simple models for the behavior of individual processes like automata. The communication primitives used to define component interaction are typically the main source of change of expressive power. Therefore, considering models with additional parameters like data, topology, etc. in the context of parameterized verification makes the problem more and more challenging.

For all the above-mentioned reasons, parameterized verification presents both theoretical and practical research challenges.

Theoretical aspects Computability and complexity issues have been studied for several classes of models and requirements, see, e.g., [4,8,26,43,64]. New theories like the theory of well-structured transition systems have been invented to provide mathematical tools to reason about parameterized and infinite-state systems. The goal of the theory of well-structured transition systems is to define decision procedures based on ideals (upward closed sets) defined over well-quasi-ordered domains and operations to manipulate them. The well-quasi-ordering requirement is needed to ensure termination of the saturation procedure needed to compute predecessors [8,43]. The theory of well-structured transition systems has been applied to several models of parameterized systems, such as

- Petri nets and Petri nets extended with reset and transfer arcs [8,19,41–43,62],
- Networks of Timed Systems [13],
- Petri nets with identifiers [54,63],
- Multiset rewriting with data [2,30],
- Broadcast protocols [40] and extensions with graph topologies [34],
- Fragments of Process algebraic languages [66], e.g., CCS [22], and π-calculus [58],
- Formal models of biological systems [9,32,33,35],
- Graph Transition Systems [18],

The theory has also been applied to systems with unbounded data structures like

- Integral Relational Automata [25],
- Lossy Channel Systems [12,24],
- Gap-order Constraint Systems [20].

Technology transfer Verification tools developed for parameterized systems are often based on technologies and methods transferred from other domains, e.g., automata as in regular model checking [5], constraint and satisfiability solvers [28,47], abstract interpretation [11], theorem proving and model checking [56,57,59]. For instance, in regular model checking sets of configurations are represented via regular languages and automata. Symbolic procedures based on automata manipulation can be applied to perform traversals of the infinite search space induced by a parameterized system. In this setting, regular languages play the role of BDDs used in Symbolic Model Checking. This idea can be generalized to any assertional language that can finitely represent infinite sets of configurations and that provides enough expressive power to encode transformations needed to computed predecessors or successors states [52]. For instance, linear arithmetic constraints can be applied to symbolically reason on decision problems like coverability for Petri nets, broadcast protocols, transfer and reset nets [29]. First-order theories, as those supported by SMT solvers, are another example of powerful assertions that can be used with specialized search strategies [53], in combination with model checking algorithms [47], and model finding procedures [55]. Decision procedures for monadic second-order logic have been applied to parameterized systems, e.g., in [48,49,56].

Abstractions Abstractions can be used to avoid or improve symbolic reasoning. Cutoff properties can be viewed as a special type of abstraction that reduces parameterized verification to finite-state model checking [15,37,38]. Other types of abstraction can be used to simplify the symbolic domain used during state exploration, see, e.g.,

- Counter Abstraction [46],
- Expand, Enlarge and Check [45],
- Abstraction Refinement for Petri Nets [44],
- Monotonic Abstraction [7,10,14],
- Environment Abstraction [27],
- View Abstraction [11],

For instance, counter abstraction can be understood as an abstraction from families of concrete systems to models like Petri nets and vector addition systems in which instead the only information maintained of the original systems is the number of occurrences of components in a given state (e.g., we use the place of a Petri Net to represent a control state of the original system). Monotonic abstraction can be understood as an abstraction from a perfect model to a lossy model, i.e., a model in which we can lose information during execution. If we recast it in the well-structured transition system terminology, monotonic abstractions amount at an overapproximation of predecessor computations via upward closed sets of configurations to retain, e.g., monotonicity and termination.

Contents of the special issue This special issue collects five contributions to provide a general view of the Parameterized Verification research field combined with technical insights into algorithms and abstractions. The presented papers provide examples of applications to specific domains like that of data protection, and connections with related fields like that of evolving databases and business process models. The papers are extended versions of presentations given in the first two editions of the Parameterized Verification Workshop, a satellite event of Concur 2014 in Rome, and of Concur 2015 in Madrid. More details on the selected papers are given in the rest of the paper.

In [31], the author gives a uniform presentation of several different models of concurrent and distributed systems with broadcast communication. Decidability and complexity results for verification of safety properties, formulated in terms of the coverability problem, are discussed for each presented model. Special attention is given to concurrent models with both broadcast communication, communication topology taken from specific classes of graphs, and data.

In [3], the authors present a parameterized verification framework based on a particular type of abstract domain called View Abstraction. The abstraction is used to represent sets of concrete configurations by means of a finite set of minimal elements with bounded size, e.g., words of size at most k for overapproximating linear configurations of finite-state processes. The concretization function induces an overapproximation computed by considering configurations that contain arbitrary combinations, e.g., shuffles of words, of the elements in the abstract state. The verification algorithm is based on an iterative deepening procedure in which the parameter, incremented at each step, is the size of the minimal elements of the view abstraction. The algorithm automatically computes cutoff points via a termination test based on inclusion of denotations of the computed abstractions.

In [61], the authors focus their attention on verification of parameterized concurrent systems that operate over local and shared variables ranging over infinite domains. The proposed approach combines different kinds of techniques such as counter abstraction, predicate abstraction, and constrained monotonic abstraction. The latter abstraction can be viewed as a refinement of abstractions based on overapproximations based on lossy semantics.

In [23], the authors establish a link between verification of infinite state systems and application domains like Evolving Databases and Business Processes. Case-centric processes are represented in the framework of Data-Centric Dynamic Systems (DCDSs). Standard correctness criteria are reformulated in this framework. Decidability fragments of the resulting languages are obtained using a number of good modeling principles and the notion of state-boundedness that makes first-order variant of μ-calculus decidable.

In [60], the authors present an application of the MCMT model checker to the validation of access control policies. The MCMT model checker is based on an SMT engine and can handle parameterized specifications as unbounded arrays. The MCMT model checker has been applied to several types of systems including mutual exclusion and fault tolerant protocols. In the paper, the authors apply the framework to model and verify instances of RBAC and ARBAC policies.

The bibliographic references collected in this introduction together with those provided in each paper cover a wide range of aspects related to parameterized verification and should help in understanding research goals and challenges for researchers interested in the field. In particular, research on parameterized verification for distributed systems and protocols, and graph-based systems seems to be still at an early stage. Furthermore, most of the positive results for parameterized verification are related to safety and reachability properties. Considering richer specification languages, e.g., temporal properties that can capture both safety and liveness, is another challenging direction for future development in the area.

References

1. Abadi, M., Blanchet, B.: Analyzing security protocols with secrecy types and logic programs. J. ACM **52**(1), 102–146 (2005)
2. Abdulla, P., Delzanno, G.: Constrained multiset rewriting. In: Proc. of the 6th international workshop on automated verification of infinite-state systems (AVIS' 2006) (2006)
3. Abdulla, P., Haziza, F., Holik, L.: Parameterized verification through view abstraction. STTT (In this issue) (2016)
4. Abdulla, P.A.: Well (and better) quasi-ordered transition systems. Bull. Symb. Logic **16**(4), 457–515 (2010)
5. Abdulla, P.A.: Regular model checking. STTT **14**(2), 109–118 (2012)
6. Abdulla, P.A., Atig, M.F., Chen, Y.-F., Leonardsson, C., Rezine, A.: Counter-example guided fence insertion under TSO. In: Tools and Algorithms for the Construction and Analysis of Systems—18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24–April 1, 2012. Proceedings, pp. 204–219 (2012)
7. Abdulla, P.A., Cederberg, J., Vojnar, T.: Monotonic abstraction for programs with multiply-linked structures. Int. J. Found. Comput. Sci. **24**(2), 187–210 (2013)
8. Abdulla, P.A., Cerans, K., Jonsson, B., Tsay, Y.-K.: General decidability theorems for infinite-state systems. In: Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, 27–30 July, 1996. IEEE Computer Society, Washington (1996)
9. Abdulla, P.A., Delzanno, G., Van Begin, L.: On the qualitative analysis of conformon P systems. In: Membrane Computing—9th International Workshop, WMC 2008, Edinburgh, UK, July 28–31, 2008, Revised Selected and Invited Papers, pp. 78–94 (2008)
10. Abdulla, P.A., Delzanno, G., Rezine, A.: Approximated parameterized verification of infinite-state processes with global conditions. Form Methods Syst. Des. **34**(2), 126–156 (2009)
11. Abdulla, P.A., Haziza, F., Holík, L.: All for the price of few. In: Verification, Model Checking, and Abstract Interpretation, 14th International Conference, VMCAI 2013, Rome, Italy, January 20–22, 2013. Proceedings, pp. 476–495 (2013)
12. Abdulla, P.A., Jonsson, B.: Undecidable verification problems for programs with unreliable channels. Inf. Comput. **130**(1), 71–90 (1996)
13. Abdulla, P.A., Jonsson, B.: Verifying networks of timed processes (extended abstract). In: Tools and Algorithms for Construction and Analysis of Systems, 4th International Conference, TACAS '98, Held as Part of the European Joint Conferences on the Theory and Practice of Software, ETAPS'98, Lisbon, Portugal, March 28—April 4, 1998, Proceedings, pp. 298–312 (1998)
14. Abdulla, P.A., Delzanno, G., Rezine, A.: Monotonic abstraction in action. In: Theoretical Aspects of Computing—ICTAC 2008, 5th International Colloquium, Istanbul, Turkey, September 1–3, 2008. Proceedings, pp. 50–65 (2008)
15. Aminof, B., Kotek, T., Rubin, S., Spegni, F., Veith, H.: Parameterized model checking of rendezvous systems. In: CONCUR 2014—Concurrency Theory—25th International Conference, CONCUR 2014, Rome, Italy, September 2–5, 2014. Proceedings, pp. 109–124 (2014)
16. Apt, K.R., Kozen, D.: Limits for automatic verification of finite-state concurrent systems. Inf. Process. Lett. **22**(6), 307–309 (1986)
17. Ball, T., Chaki, S., Rajamani, S.K.: Parameterized verification of multithreaded software libraries. In: Tools and Algorithms for the Construction and Analysis of Systems—7th International Conference, TACAS 2001, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2001, Genova, Italy, 2–6 April, 2001. Lecture Notes in Computer Science, vol. 2031, pp. 158–173. Springer, Berlin (2001)
18. Bertrand, N., Delzanno, G., König, B., Sangnier, A., Stückrath, J.: On the decidability status of reachability and coverability in graph transformation systems. In: 23rd International conference on rewriting techniques and applications (RTA' 12), RTA 2012, May 28–June 2, 2012, Nagoya, Japan. LIPIcs 15, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, pp. 101–116 (2012)
19. Bonnet, R.: The reachability problem for vector addition system with one zero-test. In: Mathematical Foundations of Computer Science 2011—36th International Symposium, MFCS 2011, Warsaw, Poland, August 22–26, 2011. Proceedings, pp. 145–157 (2011)
20. Bozzelli, L., Pinchinat, S.: Verification of gap-order constraint abstractions of counter systems. Theor. Comput. Sci. **523**, 1–36 (2014)
21. Browne, M.C., Clarke, E.M., Grumberg, O.: Reasoning about networks with many identical finite state processes. Inf. Comput. **81**(1), 13–31 (1989)
22. Busi, N., Gabbrielli, M., Zavattaro, G.: On the expressive power of recursion, replication and iteration in process calculi. Math. Struct. Comput. Sci. **19**(6), 1191–1222 (2009)
23. Montali, M., Calvanese, D.: Soundness of data-aware, case-centric processes. Int. J. Softw. Tools Technol. Transf. (2016). doi:10.1007/s10009-016-0417-2 (**In this special issue**)
24. Cécé, G., Finkel, A., Iyer, S.P.: Unreliable channels are easier to verify than perfect channels. Inf. Comput. **124**(1), 20–31 (1996)
25. Cerans, K.: Deciding properties of integral relational automata. In: Automata, Languages and Programming, 21st International Colloquium, ICALP94, Jerusalem, Israel, July 11–14, 1994, Proceedings, pp. 35–46 (1994)
26. Chambart, P., Schnoebelen, P.h.: Mixing lossy and perfect fifo channels. In: CONCUR 2008—Concurrency Theory, 19th International conference, CONCUR 2008, Toronto, Canada, 19–22 August, 2008. Lecture Notes in Computer Science, vol. 5201, pp. 340–355. Springer, Heidelberg (2008)
27. Clarke, E.M., Talupur, M., Veith, H.: Environment abstraction for parameterized verification. In: Verification, Model Checking, and Abstract Interpretation, 7th International Conference, VMCAI 2006, Charleston, SC, USA, January 8–10, 2006, Proceedings, pp. 126–141 (2006)
28. Delzanno, G.: An overview of MSR(C): a CLP-based framework for the symbolic verification of parameterized concurrent systems. Electr. Notes Theor. Comput. Sci. **76**, 65–82 (2002)
29. Delzanno, G.: Constraint-based verification of parameterized cache coherence protocols. FMSD **23**(3), 257–301 (2003)
30. Delzanno, G.: Constraint-based automatic verification of abstract models of multithreaded programs. TPLP **7**(1–2), 67–91 (2007)
31. Delzanno, G.: A unified view of parameterized verification of abstract models of broadcast communication. Int. J. Softw. Tools Technol. Transf. (2016). doi:10.1007/s10009-016-0412-7 (**In this special issue**)
32. Delzanno, G., Van Begin, L.: A biologically inspired model with fusion and clonation of membranes. In Unconventional Computing, 7th International Conference, UC 2008, Vienna, Austria, August 25–28, 2008. Proceedings, pp. 64–82 (2008)
33. Delzanno, G., Di Giusto, C., Gabbrielli, M., Laneve, C., Zavattaro, G.: The kappa-lattice: Decidability boundaries for qualitative analysis in biological languages. In: Computational Methods in Systems Biology, 7th International Conference, CMSB 2009, Bologna, Italy, August 31–September 1, 2009. Proceedings, pp. 158–172 (2009)
34. Delzanno, G., Sangnier, A., Zavattaro, G.: On the power of cliques in the parameterized verification of ad hoc networks. In: Foundations of software science and computational structures—14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS

2011, Saarbrücken, Germany, March 26–April 3, 2011. Lecture Notes in Computer Science, vol. 6604, pp. 441–455. Springer, Heidelberg (2011)
35. Delzanno, G., Zavattaro, G.: Reachability problems in bioambients. Theor. Comput. Sci. **431**, 56–74 (2012)
36. Emerson, E.A., Kahlon, V.: Exact and efficient verification of parameterized cache coherence protocols. In: Correct Hardware Design and Verification Methods, 12th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2003, L'Aquila, Italy, October 21–24, 2003, Proceedings, pp. 247–262 (2003)
37. Emerson, E.A., Namjoshi, K.S.: Reasoning about rings. In: Conference Record of POPL'95: 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, San Francisco, California, USA, January 23–25, 1995, pp. 85–94. ACM Press, San Francisco (1995)
38. Emerson, E.A., Namjoshi, K.S.: On model checking for non-deterministic infinite-state systems. In: Thirteenth Annual IEEE Symposium on Logic in Computer Science, Indianapolis, Indiana, USA, 21–24 June, 1998, pp. 70–80. IEEE Computer Society, Los Alamitos (1998)
39. Emerson, E.A., Kahlon, V.: Parameterized model checking of ring-based message passing systems. In: CSL 2004, pp. 325–339 (2004)
40. Esparza, J., Finkel, A., Mayr, R.: On the verification of broadcast protocols. In: 14th Annual IEEE Symposium on Logic in Computer Science, Trento, Italy, 2–5 July, 1999, pp. 352–359. IEEE Computer Society, Washington (1999)
41. Finkel, A., Leroux, J.: Recent and simple algorithms for petri nets. Softw. Syst. Model. **14**(2), 719–725 (2015)
42. Finkel, A., McKenzie, P., Picaronny, C.: A well-structured framework for analysing petri net extensions. Inf. Comput. **195**(1–2), 1–29 (2004)
43. Finkel, A., Schnoebelen, Ph: Well-structured transition systems everywhere!. Theor. Comput. Sci. **256**(1–2), 63–92 (2001)
44. Ganty, P., Raskin, J.-F., Van Begin, L.: From many places to few: automatic abstraction refinement for petri nets. Fundam. Inform. **88**(3), 275–305 (2008)
45. Geeraerts, G., Raskin, J.-F., Van Begin, L.: Expand, enlarge and check: new algorithms for the coverability problem of WSTS. J. Comput. Syst. Sci. **72**(1), 180–203 (2006)
46. German, S.M., Sistla, A.P.: Reasoning about systems with many processes. J. ACM **39**(3), 675–735 (1992)
47. Ghilardi, S., Ranise, S.: Backward reachability of array-based systems by SMT solving: termination and invariant synthesis. Log. Methods Comput. Sci. 6(4) (2010)
48. Henriksen, J.G., Jensen, J.L., Jørgensen, M.E., Klarlund, N., Paige, R., Rauhe, T., Sandholm, A.: Mona: Monadic second-order logic in practice. In: Tools and Algorithms for Construction and Analysis of Systems, First International Workshop, TACAS '95, Aarhus, Denmark, May 19–20, 1995, Proceedings, pp. 89–110 (1995)
49. Jensen, J.L., Jørgensen, M.E., Klarlund, N.: Monadic second-order logic for parameterized verification. Technical report, BRICS RS-94-10 (1994)
50. Kaiser, A., Kroening, D., Wahl, T.: Lost in abstraction: monotonicity in multi-threaded programs. In: CONCUR 2014—Concurrency Theory—25th International Conference, CONCUR 2014, Rome, Italy, September 2–5, 2014. Proceedings, pp. 141–155 (2014)
51. Kaiser, A., Kroening, D., Wahl, T.: A widening approach to multi-threaded program verification. ACM Trans. Program. Lang. Syst. **36**(4):14:1–14:29 (2014)
52. Kesten, Y., Maler, O., Marcus, M., Pnueli, A., Shahar, E.: Symbolic model checking with rich assertional languages. Theor. Comput. Sci. **256**(1–2), 93–112 (2001)
53. Kloos, J., Majumdar, R., Niksic, F., Piskac, R.: Incremental, inductive coverability. In: Computer Aided Verification—25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13–19, 2013. Proceedings, pp. 158–173 (2013)
54. Lazic, R., Newcomb, T., Ouaknine, J., Roscoe, A.W., Worrell, J.: Nets with tokens which carry data. Fundam. Inform. **88**(3), 251–274 (2008)
55. Lisitsa, A.: Finite reasons for safety—parameterized verification by finite model finding. J. Autom. Reason. **51**(4), 431–451 (2013)
56. Margaria, T.: Fully automatic verification and error detection for parameterized iterative sequential circuits. In: Tools and Algorithms for Construction and Analysis of Systems, Second International Workshop, TACAS '96, Passau, Germany, March 27–29, 1996, Proceedings, pp. 258–277 (1996)
57. McMillan, K.L.: Parameterized verification of the FLASH cache coherence protocol by compositional model checking. In: Correct Hardware Design and Verification Methods, 11th IFIP WG 10.5 Advanced Research Working Conference, CHARME 2001, Livingston, Scotland, UK, September 4–7, 2001, Proceedings, pp. 179–195 (2001)
58. Meyer, R.: On boundedness in depth in the pi-calculus. IFIP TCS **2008**, 477–489 (2008)
59. Pnueli, A., Xu, J., Zuck, L.D.: Liveness with (0, 1, infty)-counter abstraction. In: Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27–31, 2002, Proceedings, pp. 107–122 (2002)
60. Ranise, S., Truong, A., Traverso, R.: Parameterized model checking for security policy analysis. Int. J. Softw. Tools Technol. Transf. (2016). doi:10.1007/s10009-015-0410-1 (**In this special issue**)
61. Ganjei, Z., Rezine, A., Enes, I.P., Peng, Z.: Counting dynamically synchronizing processes. Int. J. Softw. Tools Technol. Transf. (2016). doi:10.1007/s10009-015-0411-0 (**In this special issue**)
62. Rosa-Velardo, F., de Frutos-Escrig, D.: Decidability results for restricted models of petri nets with name creation and replication. In: Applications and Theory of Petri Nets, 30th International Conference, PETRI NETS 2009, Paris, France, 22–26 June, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5606, pp. 63–82. Springer, Berlin (2009)
63. Rosa-Velardo, F., de Frutos-Escrig, D.: Decidability and complexity of petri nets with unordered data. Theor. Comput. Sci. **412**(34), 4439–4451 (2011)
64. Schnoebelen, P.: Revisiting Ackermann-hardness for lossy counter machines and reset petri nets. In: Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, 23–27 August, 2010. Proceedings. Lecture Notes in Computer Science, vol. 6281, pp. 616–628, Springer, Berlin (2010)
65. Thayer, F.J., Herzog, J.C., Guttman, J.D.: Strand spaces: proving security protocols correct. J. Comput. Secur. **7**(1), 191–230 (1999)
66. Zavattaro, G.: When to move to transfer nets—on the limits of petri nets as models for process calculi. In: Programming Languages with Applications to Biology and Security—Essays Dedicated to Pierpaolo Degano on the Occasion of His 65th Birthday, pp. 339–353 (2015)