# A False History of True Concurrency: from Petri to Tools

Javier Esparza

Institut für Informatik, Technische Universität München
Boltzmannstr. 3, 85748 Garching, Germany

**Abstract.** I briefly review the history of the unfolding approach to model checking.

*Carl Adam Petri passed away on July 2, 2010. I learnt about his death three days later, a few hours after finishing this text. He was a very profound and highly original thinker, and will be sadly missed. This note is dedicated to his memory.*

In some papers and talks, Moshe Vardi has described the history of the automata-theoretic approach to model checking, the verification technique that inspired the SPIN model checker and other tools. He traces it back to the work of theoreticians like Büchi, Prior, Trakhtenbrot and others, whose motivations were surprisingly far away from the applications that their ideas found down the line. Inspired by this work, in this note I briefly present the origins of the unfolding approach to model checking [21], a branch of the automata-theoretic approach that alleviates the state-explosion problem caused by concurrency.

Since the unfolding approach is based on the theory of *true concurrency*, describing its origins requires to speak about the origin of true concurrency itself. However, here I only touch upon those aspects of the theory that directly inspired the unfolding approach. This leaves many important works out, and so this is a very partial and "false" history of true concurrency.

The theory of true concurrency starts with two fundamental contributions by Carl Adam Petri, described in detail by Brauer and Reisig in an excellent article [11]. Both were a result of Petri's interest in analyzing the connection between mathematical, abstract computing machines, and their physical realizations. In his dissertation "Kommunikation mit Automaten", defended in 1962, Petri observes that the performance of a physically implemented Turing machine will degrade over time if the machine uses more and more storage space, because in this case signals have to travel longer and longer distances over longer and longer wires. To solve this problem he proposes an *asynchronous* architecture in which the storage space can be extended while the machine continues to operate. In the dissertation this abstract machine is described with the help of several semi-formal representations, but three years later Petri has already distilled the first mathematical formalism for asynchronous computation, and, arguably, the beginning of concurrency theory: Petri nets.

Petri's second contribution is an analysis of the notion of execution of a machine as a sequence of global states, or as a sequence of events ordered by their occurrence times with respect to some global clock. He observes that global states or global clocks are again a mathematical construct that cannot be "implemented": since information can only travel at finite speed, no part of a system can know the state of all its components at a certain moment in time.[1] He proposes to replace executions by *nonsequential processes*, sets of events ordered not by the time at which they occur, but by the *causality relation*, which is independent of the observer. The theory of nonsequential processes, subsequently developed by Goltz, Reisig, Best, Devillers, and Fernández, among others [26, 8, 9], distinguishes a system that concurrently executes two events $a$ and $b$ (usually denoted by $a \parallel b$ in process algebra) from a system that chooses between executing $a$ and then $b$, or $b$ and then $a$, (usually denoted by $a.b + b.a$): they have the same executions, namely $ab$ and $ba$, but different nonsequential processes. In fact, $a \parallel b$ is not equivalent to any sequential system, and hence the name "truly concurrent semantics" or "true concurrency."

The next important step was due to Nielsen, Plotkin, and Winskel [51] in the early 80s. Recall that the executions of a system can be bundled together into a *computation tree*: the tree of events in which the nodes are the global states, and the children of a state are the states that the system may possibly reach next (where "next" implicitly assumes a global clock). Similarly, Nielsen, Plotkin, and Winskel showed how to bundle the nonsequential processes of the system into the *unfolding* of the system, a truly-concurrent branching-time semantics.[2] Incidentally, their motivation was to extend Scott's thesis (stating that the functions between datatypes computable by sequential programs are the continuous functions) to concurrent programs. The theory of unfoldings was further developed by Engelfriet, Nielsen, Rozenberg, Thiagarajan, Winskel, and others in [52, 53, 60, 61, 18].

All this research was taking place in the area of semantics, with semantic goals: to provide a precise, formal definition of the behaviour of a concurrent system that could be used as a reference object to prove the correctness of, for instance, proof systems à la Hoare. The success of model checking introduced a new way of looking at semantics: semantical objects were not only mathematical objects that allowed to formally prove the correctness or completeness of proof systems; they could also be constructed and stored in a computer, and used to automatically check behavioural properties. More precisely, model checking suggested to construct and store an increasingly larger part of the (usually infinite) computation tree until all global states have been visited (which was bound to happen for systems with finitely many global states). By the end of the 80s model checking had already achieved significant success. However, it faced the state-explosion problem: the number of states of the system could grow very quickly

[1] This can be taken a bit further: relativity theory shows that if the parts of a system move with respect to each other there is no physical notion of a global moment in time.

[2] "Unfolding" is not the term used in [51].

as a function of the size of the system itself. One of the causes of the problem was concurrency: the number of global states of a system with $n$ concurrent components, each of them with $m$ local states, can be as large as $m^n$.

The state-explosion problem was attacked by Ken McMillan in his PhD Thesis "Symbolic Model Checking", where he famously proposed the use of Binary Decision Diagrams as a data structure for storing and manipulating sets of states. But the thesis also contains a second idea: instead of computing an initial part of the computation tree containing all global states (a *complete prefix*), McMillan suggests to construct a complete prefix *of the unfolding*. The unfolding of a concurrent system contains the same information as the computation tree, but encoded in a different way: where the computation tree represents all global states explicitly, as different nodes of a graph, the unfolding represents them implicitly, as the tuples of local states satisfying a certain condition. McMillan was the first to observe that this implicit representation provided a line of attack on the state-explosion problem, due to the smaller size of the implicit representation [45–47]. He showed how to algorithmically construct a complete prefix of the unfolding, and provided convincing experimental evidence that this approach contributed to solving the state-explosion problem. Thanks to McMillan's ideas, the unfolding moved from being a mathematical object, born out of abstract work on the nature of concurrency, into a data structure for compactly representing the set of global states of a concurrent system.

McMillan's approach, however, still faced two problems. First, while the complete prefix of the unfolding constructed by his algorithm was usually much more compact than a complete prefix of the computation tree, it could also be exponentially *bigger* in the worst case. Second, McMillan's algorithms could only check specific problems, like deadlock freedom or conformance. Both problems were overcome in the next years. Improved algorithms for constructing complete prefixes were described in [49, 22, 23, 31, 32, 35, 36, 38, 24], and extensions to (almost) arbitrary properties expressible in Linear Temporal Logic (LTL) were presented in [16, 19, 20].

Since 2000 the algorithms for constructing complete prefixes have been parallelized [33, 55] and distributed [5]. Initially developed for systems modeled as "plain" Petri nets, the unfolding approach has been extended to high-level Petri nets [37, 55], symmetrical Petri nets [17], unbounded Petri nets [1], nets with read arcs [59, 4], time Petri nets [25, 14, 15, 58],products of transition systems [22] automata communicating through queues [44], networks of timed automata [10, 12], process algebras [43], and graph transformation systems [3, 2]. It has been implemented many times [55, 56, 33, 42, 50, 29, 31, 20] and applied, among other problems, to conformance checking [48], analysis and synthesis of asynchronous circuits [39, 41, 40], monitoring and diagnose of discrete event systems [7, 6, 13, 27], and analysis of asynchronous communication protocols [44]. Two unfolders available online are Mole and PUNF, developed and maintained by Stefan Schwoon and Victor Khomenko, respectively [57, 34].

The unfolding approach to model checking is another example of how theoretical considerations about the nature of computation, and the relation between

ideal and physical machines, have evolved into a pragmatic technique for the automatic verification of concurrent systems.

## Acknowledgments

Many thanks to Stefan Schwoon for helpful suggestions.

## References

1. Parosh Aziz Abdulla, S. Purushothaman Iyer, and Aletta Nylén. Unfoldings of unbounded Petri nets. In E. Allen Emerson and A. Prasad Sistla, editors, *CAV*, volume 1855 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2000.
2. Paolo Baldan, Thomas Chatain, Stefan Haar, and Barbara König. Unfolding-based diagnosis of systems with an evolving topology. In Franck van Breugel and Marsha Chechik, editors, *CONCUR*, volume 5201 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2008.
3. Paolo Baldan, Andrea Corradini, and Barbara König. Verifying finite-state graph grammars: An unfolding-based approach. In Philippa Gardner and Nobuko Yoshida, editors, *CONCUR*, volume 3170 of *Lecture Notes in Computer Science*, pages 83–98. Springer, 2004.
4. Paolo Baldan, Andrea Corradini, Barbara König, and Stefan Schwoon. McMillan's complete prefix for contextual nets. *Transactions on Petri Nets and Other Models of Concurrency*, 1:199–220, November 2008. Volume 5100 of Lecture Notes in Computer Science.
5. Paolo Baldan, Stefan Haar, and Barbara König. Distributed unfolding of Petri nets. In Luca Aceto and Anna Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *Lecture Notes in Computer Science*, pages 126–141. Springer, 2006.
6. Albert Benveniste, Eric Fabre, Claude Jard, and Stefan Haar. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727, 2003.
7. Albert Benveniste, Stefan Haar, Eric Fabre, and Claude Jard. Distributed monitoring of concurrent and asynchronous systems. In Roberto M. Amadio and Denis Lugiez, editors, *CONCUR*, volume 2761 of *Lecture Notes in Computer Science*, pages 1–26. Springer, 2003.
8. Eike Best and Raymond R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987.
9. Eike Best and César Fernández. *Nonsequential Processes*. EATCS Monographs on Theoretical Computer Science. Springer, 1988.
10. Patricia Bouyer, Serge Haddad, and Pierre-Alain Reynier. Timed unfoldings for networks of timed automata. In Graf and Zhang [28], pages 292–306.
11. Wilfried Brauer and Wolfgang Reisig. Carl Adam Petri and "Petri nets". *Fundamental Concepts in Computer Science*, 3:129–139, 2009.
12. Franck Cassez, Thomas Chatain, and Claude Jard. Symbolic unfoldings for networks of timed automata. In Graf and Zhang [28], pages 307–321.
13. Thomas Chatain and Claude Jard. Symbolic diagnosis of partially observable concurrent systems. In David de Frutos-Escrig and Manuel Núñez, editors, *FORTE*, volume 3235 of *Lecture Notes in Computer Science*, pages 326–342. Springer, 2004.

14. Thomas Chatain and Claude Jard. Time supervision of concurrent systems using symbolic unfoldings of time Petri nets. In Paul Pettersson and Wang Yi, editors, *FORMATS*, volume 3829 of *Lecture Notes in Computer Science*, pages 196–210. Springer, 2005.
15. Thomas Chatain and Claude Jard. Complete finite prefixes of symbolic unfoldings of safe time Petri nets. In Susanna Donatelli and P. S. Thiagarajan, editors, *ICATPN*, volume 4024 of *Lecture Notes in Computer Science*, pages 125–145. Springer, 2006.
16. Jean-Michel Couvreur, Sébastien Grivet, and Denis Poitrenaud. Designing an LTL model-checker based on unfolding graphs. In Mogens Nielsen and Dan Simpson, editors, *Proc. of ICATPN 2000*, LNCS 1825. Springer, 2000.
17. Jean-Michel Couvreur, Sébastien Grivet, and Denis Poitrenaud. Unfolding of products of symmetrical Petri nets. In José Manuel Colom and Maciej Koutny, editors, *ICATPN*, volume 2075 of *Lecture Notes in Computer Science*, pages 121–143. Springer, 2001.
18. Joost Engelfriet. Branching processes of Petri nets. *Acta Informatica*, 28:575–591, 1991.
19. Javier Esparza and Keijo Heljanko. A new unfolding approach to LTL model checking. In Ugo Montanari, José D. P. Rolim, and Emo Welzl, editors, *ICALP*, volume 1853 of *Lecture Notes in Computer Science*, pages 475–486. Springer, 2000.
20. Javier Esparza and Keijo Heljanko. Implementing LTL model checking with net unfoldings. In Matthew B. Dwyer, editor, *SPIN*, volume 2057 of *Lecture Notes in Computer Science*, pages 37–56. Springer, 2001.
21. Javier Esparza and Keijo Heljanko. *Unfoldings - A Partial-Order Approach to Model Checking*. EATCS Monographs in Theoretical Computer Science. Springer-Verlag, 2008.
22. Javier Esparza and Stefan Römer. An unfolding algorithm for synchronous products of transition systems. In Jos C. M. Baeten and Sjouke Mauw, editors, *CONCUR*, volume 1664 of *Lecture Notes in Computer Science*, pages 2–20. Springer, 1999.
23. Javier Esparza, Stefan Römer, and Walter Vogler. An improvement of McMillan's unfolding algorithm. *Formal Methods in System Design*, 20(3):285–310, 2002.
24. Javier Esparza and Claus Schröter. Reachability analysis using net unfoldings. In *Proceeding of the Workshop Concurrency, Specification & Programming 2000, volume II of Informatik-Bericht 140*, pages 255–270. Humboldt-Universität zu Berlin, 2000.
25. Hans Fleischhack and Christian Stehno. Computing a finite prefix of a time Petri net. In Javier Esparza and Charles Lakos, editors, *ICATPN*, volume 2360 of *Lecture Notes in Computer Science*, pages 163–181. Springer, 2002.
26. Ursula Goltz and Wolfgang Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57(2/3):125–147, 1983.
27. B. Grabiec, L.-M. Traonouez, C. Jard, D. Lime, and O. H. Roux. Diagnosis using unfoldings of parametric time Petri nets. In *Proceedings of FORMATS 2010*, 2010. To appear.
28. Susanne Graf and Wenhui Zhang, editors. *Automated Technology for Verification and Analysis, 4th International Symposium, ATVA 2006, Beijing, China, October 23-26, 2006*, volume 4218 of *Lecture Notes in Computer Science*. Springer, 2006.
29. Bernd Grahlmann. The PEP tool. In Grumberg [30], pages 440–443.
30. Orna Grumberg, editor. *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings*, volume 1254 of *Lecture Notes in Computer Science*. Springer, 1997.

31. Keijo Heljanko. Using logic programs with stable model semantics to solve deadlock and reachability problems for 1-safe Petri nets. *Fundamenta Informaticae*, 37(3):247–268, 1999.
32. Keijo Heljanko. Model checking with finite complete prefixes is PSPACE-complete. In Palamidessi [54], pages 108–122.
33. Keijo Heljanko, Victor Khomenko, and Maciej Koutny. Parallelisation of the Petri net unfolding algorithm. In Joost-Pieter Katoen and Perdita Stevens, editors, *TACAS*, volume 2280 of *Lecture Notes in Computer Science*, pages 371–385. Springer, 2002.
34. Victor Khomenko. PUNF — Petri net unfolder. Available online at `http://homepages.cs.ncl.ac.uk/victor.khomenko/tools/`.
35. Victor Khomenko and Maciej Koutny. LP deadlock checking using partial order dependencies. In Palamidessi [54], pages 410–425.
36. Victor Khomenko and Maciej Koutny. Towards an efficient algorithm for unfolding Petri nets. In Kim Guldstrand Larsen and Mogens Nielsen, editors, *CONCUR*, volume 2154 of *Lecture Notes in Computer Science*, pages 366–380. Springer, 2001.
37. Victor Khomenko and Maciej Koutny. Branching processes of high-level Petri nets. In Hubert Garavel and John Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 458–472. Springer, 2003.
38. Victor Khomenko, Maciej Koutny, and Walter Vogler. Canonical prefixes of Petri net unfoldings. *Acta Informatica*, 40(2):95–118, 2003.
39. Victor Khomenko, Maciej Koutny, and Alexandre Yakovlev. Detecting state encoding conflicts in STG unfoldings using SAT. *Fundamenta Informaticae*, 62(2):221–241, 2004.
40. Victor Khomenko, Maciej Koutny, and Alexandre Yakovlev. Logic synthesis for asynchronous circuits based on STG unfoldings and incremental SAT. *Fundamenta Informaticae*, 70(1-2):49–73, 2006.
41. Victor Khomenko, Agnes Madalinski, and Alexandre Yakovlev. Resolution of encoding conflicts by signal insertion and concurrency reduction based on STG unfoldings. In *ACSD*, pages 57–68. IEEE Computer Society, 2006.
42. Barbara König and Vitali Kozioura. AUGUR - A tool for the analysis of graph transformation systems. *Bulletin of the EATCS*, 87:126–137, 2005.
43. Rom Langerak and Ed Brinksma. A complete finite prefix for process algebra. In Nicolas Halbwachs and Doron Peled, editors, *CAV*, volume 1633 of *Lecture Notes in Computer Science*, pages 184–195. Springer, 1999.
44. Yu Lei and S. Purushothaman Iyer. An approach to unfolding asynchronous communication protocols. In John Fitzgerald, Ian J. Hayes, and Andrzej Tarlecki, editors, *FM*, volume 3582 of *Lecture Notes in Computer Science*, pages 334–349. Springer, 2005.
45. Kenneth L. McMillan. Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits. In Gregor von Bochmann and David K. Probst, editors, *CAV*, volume 663 of *Lecture Notes in Computer Science*, pages 164–177. Springer, 1992.
46. Kenneth L. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.
47. Kenneth L. McMillan. A technique of state space search based on unfolding. *Formal Methods in System Design*, 6(1):45–65, 1995.
48. Kenneth L. McMillan. Trace theoretic verification of asynchronous circuits using unfoldings. In Pierre Wolper, editor, *CAV*, volume 939 of *Lecture Notes in Computer Science*, pages 180–195. Springer, 1995.

49. Stephan Melzer and Stefan Römer. Deadlock checking using net unfoldings. In Grumberg [30], pages 352–363.
50. Stephan Melzer, Stefan Römer, and Javier Esparza. Verification using PEP. In Martin Wirsing and Maurice Nivat, editors, *AMAST*, volume 1101 of *Lecture Notes in Computer Science*, pages 591–594. Springer, 1996.
51. Mogens Nielsen, Gordon D. Plotkin, and Glynn Winskel. Petri nets, event structures and domains. *Theoretical Computer Science*, 13(1):85–108, 1981.
52. Mogens Nielsen, Grzegorz Rozenberg, and P. S. Thiagarajan. Behavioural notions for elementary net systems. *Distributed Computing*, 4:45–57, 1990.
53. Mogens Nielsen, Grzegorz Rozenberg, and P. S. Thiagarajan. Transition systems, event structures and unfoldings. *Information and Computation*, 118(2):191–207, 1995.
54. Catuscia Palamidessi, editor. *CONCUR 2000 - Concurrency Theory, 11th International Conference, University Park, PA, USA, August 22-25, 2000, Proceedings*, volume 1877 of *Lecture Notes in Computer Science*. Springer, 2000.
55. Claus Schröter and Victor Khomenko. Parallel LTL-X model checking of high-level Petri nets based on unfoldings. In Rajeev Alur and Doron Peled, editors, *CAV*, volume 3114 of *Lecture Notes in Computer Science*, pages 109–121. Springer, 2004.
56. Claus Schröter, Stefan Schwoon, and Javier Esparza. The model-checking kit. In Wil M. P. van der Aalst and Eike Best, editors, *ICATPN*, volume 2679 of *Lecture Notes in Computer Science*, pages 463–472. Springer, 2003.
57. Stefan Schwoon. Mole — a Petri net unfolder. Available online at `http://www.lsv.ens-cachan.fr/~schwoon/tools/mole/`.
58. L.-M. Traonouez, B. Grabiec, C. Jard, D. Lime, and O. H. Roux. Symbolic unfolding of parametric stopwatch petri nets. In *Proceedings of ATVA 2010*, 2010. To appear.
59. Walter Vogler, Alexei L. Semenov, and Alexandre Yakovlev. Unfolding and finite prefix for nets with read arcs. In Davide Sangiorgi and Robert de Simone, editors, *CONCUR*, volume 1466 of *Lecture Notes in Computer Science*, pages 501–516. Springer, 1998.
60. Glynn Winskel. Event structures. In Wilfried Brauer, Wolfgang Reisig, and Grzegorz Rozenberg, editors, *Advances in Petri Nets*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.
61. Glynn Winskel. An introduction to event structures. In J. W. de Bakker, Willem P. de Roever, and Grzegorz Rozenberg, editors, *REX Workshop*, volume 354 of *Lecture Notes in Computer Science*, pages 364–397. Springer, 1988.