On-The-Fly Verification: Advancements in Dependency Graphs

Jiří Srba
Department of Computer Science, Aalborg University, Denmark

Abstract

Dependency graphs have emerged as a versatile and powerful formalism with wide-ranging applications in formal verification. In this extended abstract, we provide an overview of selected advancements in on-the-fly verification techniques based on dependency graphs, focusing on the recent developments, optimizations and generalizations of this generic verification framework.

2012 ACM Subject Classification Theory of computation

Keywords and phrases dependency graphs, Boolean equation systems, on-the-fly algorithms, fixed-point computation, applications

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2025.3

Category Invited Talk

Acknowledgements I want to thank many of my co-authors for their contributions to the development of the dependency graph framework and its applications, including J.R. Andersen, N. Andersen, A.E. Dalsgaard, S. Enevoldsen, P. Fogh, M.M. Hansen, L.S. Jensen, M.C. Jensen, N.O. Jensen, P.G. Jensen, T.S. Jepsen, I. Kaufmann, K.G. Larsen, D. Lime, A. Mariegaard, S.M. Nielsen, S.R. Olesen, M.Chr. Olesen and S. Pastva.

1 Introduction

On-the-fly model checking (also called local model checking) is an efficient approach to formal verification that explores complex systems incrementally, enabling analysis of large models without necessarily constructing their entire state-space upfront [28, 29, 33, 5, 3, 34].

Dependency graphs, introduced by Liu and Smolka [31], are directed graphs where nodes are associated with Boolean values and hyperedges connect a source node to a number of target nodes in order to represent causal dependencies in the graph. An assignment of nodes to Boolean values 0 (false) and 1 (true) is a fixed-point assignment if for every hyperedge in the graph, the source node must be assigned the value 1 whenever all of the target nodes already have the value 1. Clearly, assigning 1 to all nodes gives us a fixed-point assignment, however, we are interested in the minimum fixed-point assignment which can be obtained by initially assigning 0 to all nodes and repeatedly improving the value 0 to 1 as long as this is required by some hyperedge. Note that we allow hyperedges with an empty set of target nodes which implies that the source node must necessarily obtain the value 1 in the minimum fixed-point assignment.

We are usually only interested in the minimum fixed-point value of a specific root node, which allows us in certain situations to explore only a subset of nodes in order to determine the value of the root node. In their seminal work [31], Liu and Smolka provided an elegant, linear-time algorithm for computing the minimum fixed-point assigment of the root node in a given dependency graph where the nodes of the graph are constructed on-the-fly during a forward exploration and where the newly discovered assignments of value 1 are back-propagated along the hyperedges that are stored in a dynamically updated dependency list. Similar ideas of local algorithms were developed also for a related formalism of Boolean Equation Systems (BES) [1, 2, 30].

36th International Conference on Concurrency Theory (CONCUR 2025).
Editors: Patricia Bouyer and Jaco van de Pol; Article No. 3; pp. 3:1–3:5
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

The generic formalism of a dependency graph, together with the efficient linear-time algorithm for computing fixed points, has found numerous applications in the equivalence and model checking community. An overview of such applications can be found in [14, 16]. Among others, dependency graphs stand behind the verification engine UPPAAL TIGA for strategy synthesis in timed games [7] and are the key-stone of the CTL verification engine for Petri nets [10, 23], implemented in the tool TAPAAL [12]. The tool CAAL [4] is completely built on the dependency graph framework, supporting both bisimulation and model checking of CCS processes, including the possibility of playing games based on the generated minimum fixed-point assignment.

The dependency graph formalism was further extended towards more expressive domains for node assignments like integer domains for weighted CTL [18, 19] and even more complex domains used for PCTL model checking [32], further generalized to weighted PCTL [21] and multiplayer stochastic games [13]. Dependency graphs showed their usefulness for the synthesis problems on multi-weighted games with branching conditions [20, 27] as well as timed-arc Petri net games with discrete time [24, 25]. Recently, dependency graphs were used for the verification and synthesis of (untimed) alternating-time logic (ATL) properties in concurrent games with multiple players [6] as well as for the timed variant of ATL that allows us to specify player-coalitions [22]. Parametric systems were also analyzed by means of dependency graphs [8].

Concurrently, there has been an on-going effort to further optimize and speed-up the algorithms for computing minimum fixed-point assignments on dependency graphs and their extensions. The certain-zero technique [10] introduces a back-propagation of the Boolean value 0 along the dependent hyperedges, as soon as it becomes clear that this value cannot be further improved to 1. This often allows us to achieve an early termination of the algorithms also for the negative cases where the root node can never obtain the value 1. Negation edges, introduced in [10], enable an on-the-fly computation of mixed minimum and maximum fixed-points, as needed for example for the encoding of logical negation. The work in [15] introduces the "ignore" function that can speed-up the evaluation of nodes by determining a subset of target nodes that are still relevant for the evaluation of the source node. This allows us to prune the list of dependencies of nodes and the paper [15] discusses a recursive algorithm for implementing such a pruning approach. A more light-weight but practically beneficial variant of the pruning approach, the so-called "elimination of detached regions" in the dependency graph [26], shows promising experimental results. Recently, the technique of inclusion checking that formalizes yet another version of on-the-fly pruning of the state-space has been defined in the dependency graph framework [22], further improving the performance of the algorithms. Furthermore, there exists a line of work that describes a successful distribution of the exploration work among a number of distributed processes (workers) that communicate via message passing [11, 10, 9].

As many of the applications mentioned above include the implementation of a particular variant of the minimum fixed-point algorithm for some given abstract domain, the work on abstract dependency graphs [15] generalizes all these single-purpose implementations into a unified framework and provides a single, efficient implementation[1] that can be instantiated to many different cases as outlined in [17]. If the node assignment is taking values from the abstract algebraic structure of a Noetherian partial order with the least element, the termination of the generic fixed-point computation algorithm is guaranteed.

[1] `https://launchpad.net/adg-tool/`

References

1 Henrik Reif Andersen. Model checking and Boolean graphs. In Bernd Krieg-Brückner, editor, *ESOP '92, 4th European Symposium on Programming, Rennes, France, February 26-28, 1992, Proceedings*, volume 582 of *Lecture Notes in Computer Science*, pages 1–19. Springer, 1992. doi:10.1007/3-540-55253-7_1.

2 Henrik Reif Andersen. Model checking and Boolean graphs. *Theoretical Computer Science*, 126(1):3–30, 1994. doi:10.1016/0304-3975(94)90266-6.

3 Henrik Reif Andersen and Glynn Winskel. Compositional checking of satisfaction. In Kim Guldstrand Larsen and Arne Skou, editors, *Computer Aided Verification, 3rd International Workshop, CAV '91, Aalborg, Denmark, July, 1-4, 1991, Proceedings*, volume 575 of *Lecture Notes in Computer Science*, pages 24–36. Springer, 1991. doi:10.1007/3-540-55179-4_4.

4 J.R. Andersen, N. Andersen, S. Enevoldsen, M.M. Hansen, K.G. Larsen, S.R. Olesen, J. Srba, and J.K. Wortmann. CAAL: Concurrency workbench, Aalborg edition. In *Proceedings of the 12th International Colloquium on Theoretical Aspects of Computing (ICTAC'15)*, volume 9399 of *LNCS*, pages 573–582. Springer, 2015. doi:10.1007/978-3-319-25150-9_33.

5 Julian C. Bradfield and Colin Stirling. Local model checking for infinite state spaces. *Theor. Comput. Sci.*, 96(1):157–174, 1992. doi:10.1016/0304-3975(92)90183-G.

6 Falke B. Ø. Carlsen, Lars Bo P. Frydenskov, Nicolaj Ø. Jensen, Jener Rasmussen, Mathias M. Sørensen, Asger G. Weirsøe, Mathias C. Jensen, and Kim G. Larsen. CGAAL: Distributed on-the-fly ATL model checker with heuristics. *Electronic Proceedings in Theoretical Computer Science*, 390:99–114, September 2023. doi:10.4204/eptcs.390.7.

7 Franck Cassez, Alexandre David, Emmanuel Fleury, Kim G. Larsen, and Didier Lime. Efficient on-the-fly algorithms for the analysis of timed games. In Martín Abadi and Luca de Alfaro, editors, *CONCUR 2005 – Concurrency Theory*, volume 3653 of *LNCS*, pages 66–80, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. doi:10.1007/11539452_9.

8 Peter Christoffersen, Mikkel Hansen, Anders Mariegaard, Julian Trier Ringsmose, Kim Guldstrand Larsen, and Radu Mardare. Parametric Verification of Weighted Systems. In Étienne André and Goran Frehse, editors, *2nd International Workshop on Synthesis of Complex Parameters (SynCoP'15)*, volume 44 of *OpenAccess Series in Informatics (OASIcs)*, pages 77–90, Dagstuhl, Germany, 2015. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/OASIcs.SynCoP.2015.77.

9 A.E. Dalsgaard, S. Enevoldsen, P. Fogh, L.S. Jensen, P.G. Jensen, T.S. Jepsen, I. Kaufmann, K.G. Larsen, S.M. Nielsen, M.Chr. Olesen, S. Pastva, and J. Srba. A distributed fixed-point algorithm for extended dependency graphs. *Fundamenta Informaticae*, 161(4):351–381, 2018. doi:10.3233/FI-2018-1707.

10 A.E. Dalsgaard, S. Enevoldsen, P. Fogh, L.S. Jensen, T.S. Jepsen, I. Kaufmann, K.G. Larsen, S.M. Nielsen, M.Chr. Olesen, S. Pastva, and J. Srba. Extended dependency graphs and efficient distributed fixed-point computation. In *Proceedings of the 38th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets'17)*, volume 10258 of *LNCS*, pages 139–158. Springer-Verlag, 2017. doi:10.1007/978-3-319-57861-3_10.

11 A.E. Dalsgaard, S. Enevoldsen, K.G. Larsen, and J. Srba. Distributed computation of fixed points on dependency graphs. In *Proceedings of Symposium on Dependable Software Engineering: Theories, Tools and Applications (SETTA'16)*, volume 9984 of *LNCS*, pages 197–212. Springer, 2016. doi:10.1007/978-3-319-47677-3_13.

12 A. David, L. Jacobsen, M. Jacobsen, K.Y. Jørgensen, M.H. Møller, and J. Srba. TAPAAL 2.0: Integrated development environment for timed-arc Petri nets. In *Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, volume 7214 of *LNCS*, pages 492–497. Springer-Verlag, 2012. doi:10.1007/978-3-642-28756-5_36.

13 S. Enevoldsen, M.C. Jensen, K.G. Larsen, A. Mariegaard, and J. Srba. Verification of multiplayer stochastic games via abstract dependency graphs. In *Proceedings of the 30th*

International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR'20), volume 12561 of *LNCS*, pages 249–268. Springer, 2020. doi:10.1007/978-3-030-68446-4_13.

14 S. Enevoldsen, K.G. Larsen, A. Mariegaard, and J. Srba. Dependency graphs with applications to verification. *International Journal on Software Tools for Technology Transfer (STTT)*, 22:635–654, 2020. doi:10.1007/s10009-020-00578-9.

15 S. Enevoldsen, K.G. Larsen, and J. Srba. Abstract dependency graphs and their application to model checking. In *Proceedings of the 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'19)*, volume 11427 of *LNCS*, pages 316–333. Springer-Verlag, 2019. doi:10.1007/978-3-030-17462-0_18.

16 S. Enevoldsen, K.G. Larsen, and J. Srba. Model verification through dependency graphs. In *Proceedings of the 26th International SPIN Symposium on Model Checking of Software (SPIN'19)*, volume 11636 of *LNCS*, pages 1–19. Springer, 2019. doi:10.1007/978-3-030-30923-7_1.

17 S. Enevoldsen, K.G. Larsen, and J. Srba. Extended abstract dependency graphs. *International Journal on Software Tools for Technology Transfer (STTT)*, 24:49–65, 2022. doi:10.1007/s10009-021-00638-8.

18 J.F. Jensen, K.G. Larsen, J. Srba, and L.K. Oestergaard. Local model checking of weighted CTL with upper-bound constraints. In *Proceedings of International SPIN Symposium on Model Checking of Software (SPIN'13)*, volume 7976 of *LNCS*, pages 178–195. Springer-Verlag, 2013. doi:10.1007/978-3-642-39176-7_12.

19 J.F. Jensen, K.G. Larsen, J. Srba, and L.K. Oestergaard. Efficient model checking of weighted CTL with upper-bound constraints. *International Journal on Software Tools for Technology Transfer (STTT)*, 18(4):409–426, 2016. doi:10.1007/s10009-014-0359-5.

20 J.S. Jensen, I. Kaufmann, K.G. Larsen, S.M. Nielsen, and J. Srba. Model checking and synthesis for branching multi-weighted logics. *Journal of Logical and Algebraic Methods in Programming*, 105(1):28–46, 2019. doi:10.1016/j.jlamp.2019.02.001.

21 Mathias Claus Jensen, Anders Mariegaard, and Kim Guldstrand Larsen. Symbolic model checking of weighted PCTL using dependency graphs. In Julia M. Badger and Kristin Yvonne Rozier, editors, *NASA Formal Methods - 11th International Symposium, NFM 2019, Houston, TX, USA, May 7-9, 2019, Proceedings*, volume 11460 of *Lecture Notes in Computer Science*, pages 298–315. Springer, 2019. doi:10.1007/978-3-030-20652-9_20.

22 Nicolaj Ø. Jensen, Kim G. Larsen, Didier Lime, and Jiří Srba. On-the-fly symbolic algorithm for timed ATL with abstractions. In *36th International Conference on Concurrency Theory (CONCUR'25)*, pages 1–18. Schloss Dagstuhl, 2025. To appear.

23 N.O. Jensen, K.G. Larsen, and J. Srba. Token elimination in model checking of Petri nets. In *Proceedings of the 31st International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'25)*, volume 15696 of *LNCS*, pages 211–230. Springer-Verlag, 2025. doi:10.1007/978-3-031-90643-5_11.

24 P.G. Jensen, K.G. Larsen, and J. Srba. Real-time strategy synthesis for timed-arc Petri net games via discretization. In *Proceedings of the 23rd International SPIN Symposium on Model Checking of Software (SPIN'16)*, volume 9641 of *LNCS*, pages 129–146. Springer-Verlag, 2016. doi:10.1007/978-3-319-32582-8_9.

25 P.G. Jensen, K.G. Larsen, and J. Srba. Discrete and continuous strategies for timed-arc Petri net games. *International Journal on Software Tools for Technology Transfer (STTT)*, 20(5):529–546, 2018. doi:10.1007/s10009-017-0473-2.

26 P.G. Jensen, K.G. Larsen, J. Srba, and N.J. Ulrik. Elimination of detached regions in dependency graph verification. In *Proceedings of the 29th International SPIN Symposium on Model Checking of Software (SPIN'23)*, volume 13872 of *LNCS*, pages 163–179. Springer-Verlag, 2023. doi:10.1007/978-3-031-32157-3_9.

27 I. Kaufmann, K.G. Larsen, and J. Srba. Synthesis for multi-weighted games with branching-time winning conditions. In *Proceedings of the 41st International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets'20)*, volume 12152 of *LNCS*, pages 46–66. Springer-Verlag, 2020. doi:10.1007/978-3-030-51831-8_3.

28 Kim Guldstrand Larsen. Proof system for hennessy-milner logic with recursion. In Max Dauchet and Maurice Nivat, editors, *CAAP '88, 13th Colloquium on Trees in Algebra and Programming, Nancy, France, March 21-24, 1988, Proceedings*, volume 299 of *Lecture Notes in Computer Science*, pages 215–230. Springer, 1988. `doi:10.1007/BFb0026106`.

29 Kim Guldstrand Larsen. Proof systems for satisfiability in Hennessy-Milner logic with recursion. *Theor. Comput. Sci.*, 72(2&3):265–288, 1990. `doi:10.1016/0304-3975(90)90038-J`.

30 Xinxin Liu, C. R. Ramakrishnan, and Scott A. Smolka. Fully local and efficient evaluation of alternating fixed points. In *Proceedings of TACAS'98*, volume 1384 of *LNCS*, pages 5–19. Springer, 1998. `doi:10.1007/BFb0054161`.

31 Xinxin Liu and Scott A. Smolka. Simple linear-time algorithms for minimal fixed points (extended abstract). In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 53–66, London, UK, UK, 1998. Springer-Verlag. `doi:10.1007/BFB0055040`.

32 Anders Mariegaard and Kim Guldstrand Larsen. Symbolic dependency graphs for PCTL model-checking. In Alessandro Abate and Gilles Geeraerts, editors, *15th Internation Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'17)*, volume 10419 of *LNCS*, pages 153–169. Springer, 2017. `doi:10.1007/978-3-319-65765-3_9`.

33 Colin Stirling and David Walker. Local model checking in the modal mu-calculus. *Theor. Comput. Sci.*, 89(1):161–177, 1991. `doi:10.1016/0304-3975(90)90110-4`.

34 Glynn Winskel. A note on model checking the modal nu-calculus. *Theor. Comput. Sci.*, 83(1):157–167, 1991. `doi:10.1016/0304-3975(91)90043-2`.