International Journal on Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Formal Methods and Tools for Industrial Critical Systems

Maurice H. ter Beek · Kim G. Larsen · Dejan Ničković · Tim A.C. Willemse

Received: date / Revised version: date

Abstract Formal methods and tools have become well-established and widely applied to ensure the correctness of fundamental components of industrial critical systems in domains like railways, avionics and automotive. In this Introduction to the special issue, we outline a number of recent achievements concerning the use of formal methods and tools for the specification and verification of critical systems from a variety of industrial domains. These achievements are represented by eight properly revised and extended versions of papers that were selected from the 24th and 25th *International Conference on Formal Methods for Industrial Critical Systems* (FMICS 2019 and FMICS 2020).

Keywords Formal Methods · Tools · Critical Systems

1 Introduction

In industrial domains such as railways, avionics and automotive, critical (software) systems must comply with stringent dependability and safety requirements and standards. Therefore, formal methods and tools are commonly used for decades now when engineering such critical systems (cf., e.g., Craigen et al (1995); Clarke et al (1996); Hinchey and Bowen (1999); Woodcock et al (2009); Gnesi and Margaria (2013); Güdemann and Núñez (2017); Basile et al (2018); ter Beek et al (2018); Garavel et al (2020); Gleirscher and Marmsoler (2020); Margaria and Kiniry (2020); Ferrari and ter Beek (2022)), in particular in specific application domains (cf., e.g., Campos et al (2014); ter Beek et al (2016); Voas and Schaffer (2016); Ozay and Tabuada (2017); Weyers et al (2017); ter Beek and Loreti (2018); Bonfanti et al (2018); Marko et al (2020); Michael et al (2021)).

Formal methods are rigorous and mathematics-based specification languages to describe (model) system behaviour (cf., e.g., Wing (1990); Hinchey et al (2010); Almeida et al (2011); Bowen and Hinchey (2014); Nielson and Nielson (2019)) that come with a precise semantics and with tools for automated formal verification (analysis) of these system models, based on techniques like theorem proving (cf. Robinson and Voronkov (2001)) and model checking (cf. Clarke et al (2018)), including probabilistic (cf. Baier and Katoen (2008)) and statistical (cf. Agha and Palmskog (2018)) approaches.

As in other engineering disciplines, the envisioned advantage of using formal methods and tools is the expectation that appropriate mathematical modelling and analysis contributes to the correctness of the developed systems by eliminating errors in the initial designs, i.e. well before implementation, and by guaranteeing robust and fault-tolerant systems that behave as specified even in uncertain environments.

This special issue dedicated to "Formal Methods and Tools for Industrial Critical Systems" contains a total of eight papers that concern properly revised and extended versions of papers from the 24th and 25th *International Conference on Formal Methods for Industrial Critical Systems* (FMICS 2019 and FMICS 2020).

M.H. ter Beek
ISTI–CNR, Pisa, Italy
E-mail: maurice.terbeek@isti.cnr.it

K.G. Larsen
Aalborg University, Aalborg, Denmark
E-mail: kgl@cs.aau.dk

D. Ničković
AIT Austrian Institute of Technology, Vienna, Austria
E-mail: dejan.nickovic@ait.ac.at

T.A.C. Willemse
Eindhoven University of Technology, Eindhoven, Netherlands
E-mail: t.a.c.willemse@tue.nl

Based on the original reviews and the subsequent discussions among the PC members of FMICS 2019 and FMICS 2020, the PC chairs of FMICS 2019 and FMICS 2020 invited the authors of 11 papers to submit a revised and substantially extended version of their original conference paper to this special issue. The authors of nine papers decided to accept this invitation and based on a thorough reviewing process, by which each paper was reviewed by three reviewers of which at least two had not previously reviewed the conference version, the editors of this special issue decided to reject one paper and accept eight papers for inclusion in this special issue.

2 FMICS

The aim of the FMICS conference series is to provide a forum for researchers who are interested in the development and application of formal methods in industry. In particular, FMICS brings together scientists and engineers who are active in the area of formal methods and interested in exchanging their experiences in the industrial usage of these methods. FMICS also strives to promote research and development for the improvement of formal methods and tools for industrial applications. FMICS is the annual conference of the ERCIM Working Group on Formal Methods for Industrial Critical Systems[1], and it is the key conference in the intersection of industrial applications and Formal Methods.

FMICS 2019 was organised on 30 and 31 August, 2019, in Amsterdam, The Netherlands. FMICS 2020 was organised on 2 and 3 September, 2020, in Vienna, Austria. Both called for contributions on the following, non-exhaustive, topics of interest:

- Case studies and experience reports on industrial applications of formal methods, focusing on lessons learned or identification of new research directions.
- Methods, techniques and tools to support automated analysis, certification, debugging, descriptions, learning, optimisation and transformation of complex, distributed, real-time, embedded, mobile and autonomous systems.
- Verification and validation methods (model checking, theorem proving, SAT/SMT constraint solving, abstract interpretation, etc.) that address shortcomings of existing methods with respect to their industrial applicability (e.g., scalability and usability issues).
- Impact of the adoption of formal methods on the development process and associated costs. Application of formal methods in standardisation and industrial forums.

The proceedings of both FMICS 2019 and FMICS 2020 have been published in Springer's Lecture Notes in Computer Science series (cf. Larsen and Willemse (2019); ter Beek and Ničković (2020)).

3 Selected Papers

In the remainder of this Introduction to the special issue, we briefly present and contextualise the contributions of the papers that make up this special issue, followed by a brief discussion of the overall impact of this special issue.

3.1 Railways

Shift2Rail and its successor Europe's Rail[2] are joint undertakings aimed to increase the competitiveness of the European railway industry. This concerns in particular the transition to the next generation of EU signalling systems, which will include satellite-based train positioning, moving-block distancing and automatic driving. A further aim is to contribute to the EU strategy for more sustainable and smart mobility by funding the delivery of more flexible approaches to planning and traffic management of rail services, which will increase capacity and support smart and cost-efficient rail connectivity. The railway domain is a field in which formal methods and tools are traditionally applied successfully (cf., e.g., Ferrari and ter Beek (2022)).

The paper *DFT Modeling Approach for Operational Risk Assessment of Railway Infrastructure*, by Weik et al (2022), the recipient of the FMICS 2019 Best Paper Award, describes a fully automated approach to model railway station areas and train routability using dynamic fault trees (DFTs). Their approach allows for quantitatively analysing the reliability of train operations, offering insights into the reliability of the train station area and the criticality of wayside equipment such as switches, signals, or train detection systems, with respect to the routing possibilities. For the transformations and analysis the authors use the Storm framework[3]. The authors first simplify the DFTs by rewriting, thereby, for instance, removing redundant levels of ANDs or ORs, and subsequently use Storm to generate a continuous-time Markov chain (CTMC) capturing the behaviour of the DFT. These CTMCs are then analysed for a selection of quality metrics, which

[1] https://fmics.inria.fr/

[2] https://shift2rail.org/

[3] https://www.stormchecker.org/

are specified in Continuous Stochastic Logic with reward extensions. To assess their approach, the authors analyse 16 DFTs originating from four German railway stations, three of which are major central stations (up to 10 platform tracks) with multiple starting and ending lines. The fourth railway station is of medium size (4 platform tracks) but has a small freight yard. While most of the DFTs can be analysed within seconds, some of the DFTs with alternative routes require several minutes. Computing all criticalities, however, can require more than 3 days if run sequentially; the authors do note that the problem of computing all criticalities is embarrassingly parallelisable. According to the authors, their approach may, among others, be useful for a comparative assessment of wayside infrastructure elements, so as to better understand their role and influence on operations.

The paper *Exploring the ERTMS/ETCS full moving block specification: An experience with formal methods* by Basile et al (2022), presents the authors' experience with modelling and analysing scenarios that are offered by an operational model specified in Uppaal[4] of a single railway track with multiple trains that concurrently communicate with the same (trackside) radio block centre (RBC). The models are based on specifications and models developed in H2020 projects funded under the European Shift2Rail initiative of full moving block railway signalling systems that match level 3 of the European Rail Traffic Management System (ERTMS) standard. The authors apply statistical model checking with Uppaal SMC by running a sufficient number of probabilistic simulations of their system model to obtain statistical evidence (with a predefined level of statistical confidence) of the quantitative properties to be checked. In particular, they verify the correctness of a function, formalised in first-order logic, that dynamically computes the main task of the RBC, namely continuously sending to trains their movement authority (MA), which is essentially the maximum distance and speed a train is allowed to travel at. The MA is based on the traffic on the railway track and in particular on the tail of the train ahead. The authors tune and validate the physical behaviour of the trains according to parameters concerning high-speed trains taken from the literature. Their formal analysis shows the need for novel requirements to deal with several corner cases concerning start-up operations, message loss, and concurrency issues, and future research goals to improve the formal specification and verification of real-time systems. Finally, based on their experience, the authors discuss some barriers concerning a possible uptake of formal methods and tools in the railway industry.

[4] `https://uppaal.org/`

3.2 Avionics

The paper *Envelopes and Waves: Safe Multivehicle Collision Avoidance for Horizontal Non-deterministic Turns* by Kouskoulas et al (2022), studies the problem of horizontal terms, frequently encountered in autonomous systems that control aircraft. The problem is motivated by aircraft collision avoidance manoeuvres that combine vertical and horizontal advice to ensure that multi-aircraft encounters are safely separated. The paper first develops a formalisation of turn-to-bearing manoeuvres, in which an aircraft turns following a circular arc until it reaches a certain bearing, and then follows a straight path. The parameters that describe paths are assumed to be non-deterministic and uncertain at the beginning of the turn. In the next step, the authors develop a library containing geometric properties of turn-to-bearing trajectories, formalised and proved in the Coq proof assistant[5]. This library provides foundations and basic building blocks for safety analysis of horizontal turns. The paper uses the turn-to-bearing library to formally verify a non-trivial collision timing property of such trajectories and to develop a more general approach to quantifying the timing computations over a conflict area. The paper also provides sound approximations for the location of a vehicle within the reachable area and a simple and efficient procedure for finding the range of possible collision times between two vehicles that are both followings turn-to-bearing kinematics.

3.3 Automotive

The advent of autonomous vehicles imposes fundamental paradigm shifts in the automotive industry, bringing new exciting challenges (cf., e.g., Marko et al (2020)).

The paper *Verifiable Strategy Synthesis for Multiple Autonomous Agents: A Scalable Approach* by Gu et al (2022), the recipient of the FMICS 2020 Best Paper Award, addresses two major problems associated to the autonomous operation of agents: path planning and task scheduling. The joint automated solution to these two problems is also known as mission plan synthesis. The paper first develops an exhaustive method for synthesising mission plans based on model checking. The authors use *timed games* to model the task scheduling problem for an agent operating in an uncertain environment. The task scheduling problem is solved using the Uppaal Tiga timed game synthesis. This implementation is integrated into the TAMAA (Timed-Automata-based Mission planner for Autonomous Agents) tool. While exhaustive and provably correct, the TAMAA

[5] `https://coq.inria.fr/`

solution has limited scalability. Consequently, the authors propose an alternative and lighter approach to task scheduling, which combines reinforcement learning with simulation-based synthesis. Instead of symbolic state exploration, this method uses a simulation-based synthesis algorithm based on Q-learning. It samples the state space via random simulations and learns an intermediate solution to the task scheduling problem. An intermediary strategy does not necessarily cover all the situations allowed by the unpredictable environment. Hence, the synthesised strategy is then verified using Uppaal Stratego and the learning process is repeated until a correct strategy is inferred. Both task scheduling solutions are integrated into the mission planning synthesis tool MALTA. The methods are evaluated on an industrial autonomous quarry case study.

The paper *Formal Modeling and Verification for Amplification Timing Anomalies in the Superscalar TriCore Architecture* by Binder et al (2022), addresses the problem of the worst-case timing analysis of the TriCore architecture used in automotive applications. It is a sophisticated dual-pipelined superscalar processor architecture that is likely suffering from amplification timing anomalies. An amplification timing anomaly manifests itself when a local timing variation between two executions of a program results in a larger global timing variation. The paper adapts an existing canonical single pipeline model, used to study amplification timing anomalies in time-predictable processor pipelines to accommodate the specific TriCore features. The pipeline model extension is not trivial—it requires both structural modifications due to the dual pipeline, and functional adaptations to the TriCore architecture, including its progression logic, store buffer, and data dependencies. The amplification timing anomalies in TriCore are then studied using a verification procedure that uses a bounded model checking (BMC) procedure implemented on top of a satisfiability-modulo-theories (SMT) solver. The experiments consist of two steps. The authors empirically validate the formal model by testing its compliance with the pipeline description in the documented examples. The verification procedure is then used to detect amplification timing anomalies in the TriCore architecture. Finally, the paper develops a procedure based on SMT heuristics to guide the verification engine towards obtaining multiple counterexamples that exhibit amplification timing anomalies.

3.4 Formal Methods and Tools

The paper *Formal Verification of OIL Component Specifications using mCRL2* by Bunte et al (2022) presents a formal operational semantics of OIL (Open Interaction Language), a language for modelling control software, developed at Canon Production Printing, and a translation from OIL to mCRL2, along with a proof of correctness. OIL, which the authors introduce via a running example of an overheating printer, is a language with both a textual and a graphical syntax. An OIL specification consists of areas of different flavours and transitions connecting these. A distinguishing feature of OIL is that it facilitates a "constraint oriented" programming style, which allows the programmer to focus on separate aspects (concerns) of the program without having to take other aspects into account. The translation of OIL to mCRL2 opens up the possibility to analyse OIL specifications using the mCRL2 toolset[6]. In this way, arbitrary requirements, phrased in the modal μ-calculus, can be verified. Moreover, the authors formalise several generic requirements, related to the predictability and reliability of the behaviour of OIL specifications, that must be met by all OIL specifications. They show that also these requirements can be verified using the mCRL2 toolset. Since OIL specifications can be converted automatically to executable code, the ability to analyse specifications before they end up as code in products is expected to significantly improve the software and product quality. The authors report on their experience analysing two industrial, confidential OIL specifications. Finally, the authors reflect on their translation to mCRL2 and on some of the challenges that still lie ahead. One aspect the authors allude to is the fact that their translation of OIL to mCRL2 is rather monolithic, a more compositional translation being hampered by the lack of shared variables in mCRL2. They observe that this monolithic translation gives rise to complex data types which adversely affect the effectiveness of many of the tools of mCRL2, showing the need for either a more compositional translation or improvements in the mCRL2 toolset to better deal with complex data types.

The paper *Temporal-Logic Query Checking over Finite Data Streams* by Huang and Cleaveland (2022) addresses the *specification mining* problem, which consists in inferring high-level properties of a black-box system from observing its executions. In this work, system executions are finite data streams, i.e. finite sequences of state observations, where a state is a tuple consisting of a time index and a set of atomic propositions that hold in that state. The target specification language is Finite Linear Temporal Logic (Finite LTL), an LTL variant interpreted over finite traces. The authors take a template-based mining approach, in which the user provides a Finite LTL *query*—a Finite LTL specification

[6] `https://www.mcrl2.org/`

with a missing propositional subformula. In this case, specification mining corresponds to solving a *query* for Finite LTL. Solving a Finite LTL query consists in inferring all propositional formulas that, when substituted for the unknown variable in the query, results in a Finite LTL specification satisfied by all data streams in the given set. The authors adopt an automata-based approach to solving Finite LTL queries. A Finite LTL query is first translated to a a finite query automaton (FQA), which also contains the propositional query variable. The FQA is then composed with automata derived from the set of finite data stream and the solutions to the queries are computed from the composite automaton. The proposed approach is implemented and evaluated on synthetic datasets, demonstrating the meaningfulness of the solutions and the computational performance of the algorithm.

The final paper *DivSIM, an Interactive Simulator for LLVM Bitcode* by Ročkai and Barnat (2022) discusses a simulator for LLVM bitcode. This simulator, called DivSIM, is implemented in DIVINE 4, an explicit-state model checker[7] which is based on the LLVM toolchain. The main goal of this simulator is to make the analysis of complex, multi-threaded (C/C++) programs more accessible. It can be used to interactively explore traces of a program; for instance, traces that constitute a counterexample produced by a verification effort. The simulator can step through a program's execution both forwards and backwards, and it provides control over thread interleaving of parallel programs. It supports symbolic evaluation through abstraction and symbolic execution; it interprets the program, allowing the user to fix non-deterministic choices instead of relying on the scheduler from the operating system to resolve choices. DivSIM allows to make persistent snapshots of the current state, offering insight into pointer relations, but also typing information of stored values and relations between values in memory locations. The authors evaluate their simulator on a large number of test cases consisting of C and C++ programs, demonstrating robustness of their tool and its ability to replay counterexamples reported by the model checker.

4 Discussion

We have briefly presented the eight selected papers that constitute this special issue. The topics addressed in these papers cover a broad range of formal methods and tools, ranging from fault trees and automata models to theorem proving with Coq and model checking and synthesis with the mCRL2 and Uppaal toolsets. Moreover, they contain applications to critical systems from industrial domains such as railways, avionics and automotive. For future and more effective application in industry, the approaches described in the papers constituting this special issue require further development and implementation.

In the future, Weik et al (2022) would like to improve the predictive quality of their DFT models for failure processes and reliability assessment to extend the application area of their methodology beyond comparative analysis of infrastructure components to usage for requirement-based system design. Basile et al (2022) have distilled future research lines to improve the formal specification and verification of real-time systems in light of current barriers concerning their possible uptake in the railway industry. These concern in particular the level of abstraction, separation of concerns, deployment, and how to integrate tools like Uppaal in the standardised railway development process.

Kouskoulas et al (2022) plan to synthesise a safety controller and develop a formalisation and proofs of correctness for controller synthesis, which would allow them to obtain a correct-by-construction controller implementation, and for representing position uncertainty in the vehicles, which would allow them to model sensor errors or unexpected variations in trajectories.

To synthesise path planning and task scheduling strategies for multiple autonomous agents that perform better in environments with large numbers of milestones and tasks, Gu et al (2022) would like to improve the learning algorithm of their method. Binder et al (2022) intend to improve their strategies for deriving execution patterns that are likely to exhibit undesired timing phenomena, known as amplification timing anomalies, by applying them on more concrete models equipped with countermeasures that limit the occurrence of such anomalies. They eventually would like to integrate the result in worst-case execution time (WCET) analysers.

Bunte et al (2022) foresee that for the successful analysis of OIL specifications with the mCRL2 toolset in practice, they first need to hide the complexities of using mCRL2 from the engineer, for instance by offering push-button validation of requirements that can be expressed in a language that is simpler than the μ-calculus. To apply their query-solving approach to a larger scope of problems, Huang and Cleaveland (2022) need to do more experimentation, including the study of query languages that are tolerant of the noise that is typically found in many experimental data sets. Ročkai and Barnat (2022), finally, hope that the fact that DivSIM can be combined with existing LLVM-based tools for languages like Objective C or Rust and the simple, compact and universal counterexample format pro-

[7] `https://divine.fi.muni.cz/`

duced by DiVM (the DIVINE virtual machine) that can be loaded, simulated and analysed by DivSIM, will foster tool interoperability, eventually resulting in an ecosystem of tools that use DivSIM internally and cooperate through a common input format.

Acknowledgements We would like to thank all authors for their contributions to this special issue and the reviewers of FMICS 2019 and FMICS 2020, and in particular those of this special issue, for their reviews.

References

Agha G, Palmskog K (2018) A survey of statistical model checking. ACM Transactions on Modeling and Computer Simulation 28(1):6:1–6:39, doi:10.1145/3158668

Almeida JB, Frade MJ, Pinto JS, Melo de Sousa S (2011) An overview of formal methods tools and techniques. In: Rigorous Software Development: An Introduction to Program Verification, Springer, pp 15–44, doi:10.1007/978-0-85729-018-2_2

Baier C, Katoen JP (2008) Principles of Model Checking. MIT Press, Cambridge, URL `http://mitpress.mit.edu/books/principles-model-checking`

Basile D, ter Beek MH, Fantechi A, Gnesi S, Mazzanti F, Piattino A, Trentini D, Ferrari A (2018) On the industrial uptake of formal methods in the railway domain. In: Furia CA, Winter K (eds) Proceedings of the 14th International Conference on Integrated Formal Methods (iFM 2018), Springer, Lecture Notes in Computer Science, vol 11023, pp 20–29, doi:10.1007/978-3-319-98938-9_2

Basile D, ter Beek MH, Ferrari A, Legay A (2022) Exploring the ERTMS/ETCS full moving block specification: An experience with formal methods. International Journal on Software Tools for Technology Transfer In this issue

ter Beek MH, Loreti M (2018) Guest editorial for the special issue on FORmal methods for the quantitative Evaluation of Collective Adaptive SysTems (FORECAST). ACM Transactions on Modeling and Computer Simulation 28(2):8:1–8:4, doi:10.1145/3177772

ter Beek MH, Ničković D (eds) (2020) Proceedings of the 25th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2020), Lecture Notes in Computer Science, vol 12327, Springer, doi:10.1007/978-3-030-58298-2

ter Beek MH, Clarke D, Schaefer I (2016) Editorial preface for the JLAMP special issue on Formal Methods for Software Product Line Engineering. Journal of Logical and Algebraic Methods in Programming 85(1):123–124, doi:10.1016/j.jlamp.2015.09.006

ter Beek MH, Gnesi S, Knapp A (2018) Formal methods for transport systems. International Journal on Software Tools for Technology Transfer 20(3):237–241, doi:10.1007/s10009-018-0487-4

Binder B, Asavoae M, Brandner F, Ben Hedia B, Jan M (2022) Formal Modeling and Verification for Amplification Timing Anomalies in the Superscalar TriCore Architecture. International Journal on Software Tools for Technology Transfer In this issue

Bonfanti S, Gargantini A, Mashkoor A (2018) A systematic literature review of the use of formal methods in medical software systems. Journal of Software: Evolution and Process 30(5):e1943:1–e1943:18, doi:10.1002/smr.1943

Bowen JP, Hinchey MG (2014) Formal Methods. In: Gonzalez TF, Diaz-Herrera J, Tucker A (eds) Computing Handbook, CRC Press, chap 71, pp 71–25

Bunte O, van Gool LCM, Willemse TAC (2022) Formal Verification of OIL Component Specifications using mCRL2. International Journal on Software Tools for Technology Transfer In this issue

Campos J, Seatzu C, Xie X (eds) (2014) Formal Methods in Manufacturing. CRC, doi:10.1201/9781315216140

Clarke EM, Wing JM, et al (1996) Formal methods: State of the art and future directions. ACM Computing Surveys 28(4):626–643, doi:10.1145/242223.242257

Clarke EM, Henzinger TA, Veith H, Bloem R (eds) (2018) Handbook of Model Checking. Springer, doi:10.1007/978-3-319-10575-8

Craigen D, Gerhart S, Ralston T (1995) Industrial Applications of Formal Methods to Model, Design and Analyze Computer Systems: An International Survey. Advanced Computing and Telecommunication Series, William Andrew, doi:10.1016/C2009-0-20452-1

Ferrari A, ter Beek MH (2022) Formal methods in railways: a systematic mapping study. ACM Computing Surveys doi:10.1145/3520480

Garavel H, ter Beek MH, van de Pol J (2020) The 2020 expert survey on formal methods. In: ter Beek MH, Ničković D (eds) Proceedings of the 25th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2020), Springer, Lecture Notes in Computer Science, vol 12327, pp 3–69, doi:10.1007/978-3-030-58298-2_1

Gleirscher M, Marmsoler D (2020) Formal methods in dependable systems engineering: a survey of professionals from Europe and North America. Empirical Software Engineering 25(6):4473–4546, doi:10.1007/s10664-020-09836-5

Gnesi S, Margaria T (eds) (2013) Formal Methods for Industrial Critical Systems: A Survey of Applications. John Wiley & Sons, Inc., Hoboken

Gu R, Jensen PG, Poulsen DB, Seceleanu C, Enoiu E, Lundqvist K (2022) Verifiable Strategy Synthesis for Multiple Autonomous Agents: A Scalable Approach. International Journal on Software Tools for Technology Transfer In this issue

Güdemann M, Núñez M (2017) Preface of the special issue on formal methods in industrial critical systems. International Journal on Software Tools for Technology Transfer 19(4):391–393, doi:10.1007/s10009-017-0455-4

Hinchey M, Bowen JP, Vassev E (2010) Formal methods. In: Laplante PA (ed) Encyclopedia of Software Engineering, Taylor & Francis, pp 308–320, URL `http://www.crcnetbase.com/doi/abs/10.1081/E-ESE-120044313`

Hinchey MG, Bowen JP (eds) (1999) Industrial-Strength Formal Methods In Practice. Formal Approaches to Computing Information Technology, Springer, doi:10.1007/978-1-4471-0523-7

Huang S, Cleaveland R (2022) Temporal-Logic Query Checking over Finite Data Streams. International Journal on Software Tools for Technology Transfer In this issue

Kouskoulas Y, Machado TJ, Genin D, Schmidt A, Papusha I, Brulé J (2022) Envelopes and Waves: Safe Multivehicle Collision Avoidance for Horizontal Non-deterministic Turns. International Journal on Software Tools for Technology Transfer In this issue

Larsen KG, Willemse T (eds) (2019) Proceedings of the 24th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2019), Lecture Notes in Computer Science, vol 11687, Springer, doi:10.1007/978-3-030-27008-7

Margaria T, Kiniry J (2020) Welcome to formal methods in industry. IT Professional 22(1):9–12, doi:10.1109/MITP.2020.2968715

Marko N, Möhlmann E, Ničković D, Niehaus J, Priller P, Rooker M (2020) Challenges of engineering safe and secure highly automated vehicles: Whitepaper. arXiv 2103.03544 [cs.AI], URL `https://arxiv.org/abs/2103.03544`

Michael JB, Drusinsky D, Wijesekera D (2021) Formal methods in cyberphysical systems. IEEE Computer 54(9):25–29, doi:10.1109/MC.2021.3089267

Nielson F, Nielson HR (2019) Formal Methods: An Appetizer. Springer, doi:10.1007/978-3-030-05156-3

Ozay N, Tabuada P (2017) Guest editorial: special issue on formal methods in control. Discrete Event Dynamic Systems 27(2):205–208, doi:10.1007/s10626-017-0246-9

Robinson JA, Voronkov A (eds) (2001) Handbook of Automated Reasoning. Elsevier

Ročkai P, Barnat J (2022) DivSIM, an Interactive Simulator for LLVM Bitcode. International Journal on Software Tools for Technology Transfer In this issue

Voas JM, Schaffer K (2016) Insights on formal methods in cybersecurity. IEEE Computer 49(5):102–105, doi:10.1109/MC.2016.131

Weik N, Volk M, Katoen JP, Nießen N (2022) DFT Modeling Approach for Operational Risk Assessment of Railway Infrastructure. International Journal on Software Tools for Technology Transfer In this issue

Weyers B, Bowen J, Dix A, Palanque P (eds) (2017) The Handbook of Formal Methods in Human-Computer Interaction. Human–Computer Interaction Series, Springer, doi:10.1007/978-3-319-51838-1

Wing JM (1990) A specifier's introduction to formal methods. IEEE Comput 23(9):8–24, doi:10.1109/2.58215

Woodcock J, Larsen PG, Bicarregui J, Fitzgerald JS (2009) Formal methods: Practice and experience. ACM Computing Surveys 41(4):19:1–19:36, doi:10.1145/1592434.1592436