# Verification and Validation of Concurrent and Distributed Heterogeneous Systems (Track Summary)

Marieke Huisman[1] and Cristina Seceleanu[2(✉)]

[1] University of Twente, Enschede, The Netherlands
m.huisman@utwente.nl
[2] Mälardalen University, Västerås, Sweden
cristina.seceleanu@mdu.se

**Abstract.** A major trend in computing during the last decade has been the ubiquity of distributed, heterogeneous systems that make use of multi-languages for implementation, or services delivered by IoT devices. Since all distributed systems must, by their very nature, make use of some form of concurrent programming, the latter becomes even more challenging than traditionally, with the increase of hardware concurrency and sources of heterogeneity. In this context, developing reliable, safe and secure distributed systems, without sacrificing performance, is notoriously difficult. This requires novel models, logical notations, and verification techniques, or extensions, improvements and combinations of existing ones, to capture the behavior of such systems and provide guarantees that they meet various specific requirements. The track on Verification and Validation of Concurrent and Distributed Heterogeneous Systems aims to present and discuss advances of formal methods applicable to the assurance of different kinds of heterogeneous systems, as well as new insights provided by validating the methods on real-world case studies.

## 1 Motivation and Goals

Demands for computing power and better performance are only going to increase. The cloud, mobile and IoT systems are all playing a role in creating big expectations for enterprises, placing tremendous strain on developers that must manage a number of important—often conflicting—system properties, including reliability, predictability, security and others, as well as complex interactions among system components. This justifies the proliferation of frameworks and tools intended to make the design and implementation of such heterogeneous and distributed systems easier (e.g., Hadoop: https://hadoop.apache.org/docs/current/, OpenCL: https://www.khronos.org/opencl/). Nevertheless, ensuring that the latter are safe, and that their behavior remains predictable and correct, lies on the "shoulders" of formal methods, via different tools and techniques to reason about (heterogeneous) distributed systems [4,8,12,17] and concurrent software [2,5,9,11,14,16].


T. Margaria and B. Steffen (Eds.): ISoLA 2022, LNCS 13701, pp. 417–421, 2022.
https://doi.org/10.1007/978-3-031-19849-6_24

Despite many existing results, questions such as: How can formal verification techniques for concurrent systems be extended to ensure properties over distributed objects?, How can one analyze multi-lingual distributed programs efficiently?, How can one ensure properties of complex industrial concurrent software?, or How can we capture, control and verify smart applications formally? are still awaiting answers.

The **Verification and Validation of Concurrent and Distributed Heterogeneous Systems** (VVCDHS) track focuses on tackling the above questions, by presenting invited papers that propose models, techniques, and tools for the rigorous description and analysis of concurrent and distributed systems characterized by heterogeneity stemming from various sources, such as multi-language programming, the need for adaptivity, the use of smart components, or IoT applications, etc. The included contributions give a good overview of the current state-of-the-art in the verification of heterogeneous concurrent and distributed systems, and propose solutions to difficult problems related to logics for distributed objects that evolve over time, modeling and verification of systems that need to perform adaptations, modular verification of concurrent algorithms, applying academic formal verification tools to industrial systems, runtime verification of IoT device security of communication, and formal modeling of systems-of-systems control. The track closes with a discussion on ways in which we can leverage verification and validation techniques for concurrency and distribution to tackle modern topics of heterogeneous and smart systems assurance, including the scalability and industrial applicability of the formal approaches.

Finally, we would like to express our deep gratitude to the ISoLA organizers, in particular Prof. Tiziana Margaria and Prof. Bernhard Steffen, for their tenacious work to provide the infrastructure for our and other tracks, which allows vivid and creative discussions among researchers and practitioners in different communities, leading to new insights and perspectives of the techniques and foundations of formal methods and their application.

## 2 Overview of Contributions

In *An Efficient VCGen-based Modular Verification of Relational Properties* [1], the authors Lionel Blatter, Nikolai Kosmatov, Virgile Prevosto, and Pascale Le Gall propose a verification conditions generation (VCG) technique to prove relational properties over a language with support for procedure calls and aliasing. The technique enables a modular approach to verification, allowing to use relational properties as hypothesis in proving other relational properties. The language is formalized in COQ, where the authors prove the main result, that is, the verification conditions generated by the proposed VCG can be used to prove relational properties, modularly.

In *On Binding in the Spatial Logic for Closure Spaces* [3], the authors Laura Bussi, Vincenzo Ciancia, Fabio Gadducci, Diego Latella, and Mieke Massink present a logic to express properties over objects distributed over space, and

evolving over time. The authors propose two different extensions of the spatial logic for closure spaces, and its spatio-temporal variant, with spatial quantification operators useful for reasoning about the existence of particular spatial objects in a space, and the dynamic evolution of such spatial objects in time and space. The expressiveness of the operators is illustrated via several representative examples, out of which the leader election one shows that such logics could even be useful to specify the correctness of distributed algorithms.

In *An IoT Digital Twin for Cyber-Security Defence based on Runtime Verification* [6], the authors Jorge David de Hoz Diego, Anastasios Temperekidis, Panagiotis Katsaros, and Charalambos Konstantinou propose a security-decoupling solution for IoT networked devices. The proposed solution deploys a security module to the IoT device, and uses a Digital Twin (DT) to detect potentially compromised packets communicated from either a remote device or a local process. The approach provides information that one can use to mitigate potential cyber-security attacks. The rule-based runtime verification implemented under a security context assumption shows that, by isolating local traffic, the DT monitor can detect and track threats occurring by unauthorized communication attempts.

In *A thread-safe Term Library* [7], the authors Jan Friso Groote, Maurice Laveaux, and P.H.M. van Spaendonck present the design, implementation and model checking of a new thread-safe Term library with several novel aspects including the new busy-forbidden protocol, that is, a protocol with functionality similar to the readers-writers lock, but with improvements aiming to make it faster. The authors prove by model checking the key properties of the library, which guarantee thread safety and liveness.

In *ST4MP: A Blueprint of Multiparty Session Typing for Multilingual Programming* [10], the authors Sung-Shik Jongmans and José Proença provide a multiparty session types (MPST) method to simplify the construction and analysis of distributed systems implemented in multiple programming languages. In order to verify processes in such a heterogeneous setting, the authors generate API in multiple languages, by their newly introduced tool called ST4MP "Session Types Fo(u)r Multilingual Programming". The ST4MP approach so far focuses on three languages: Java, Scala and Rust, which are widely used for programming distributed systems.

In *On Deductive Verification of an Industrial Concurrent Software Component with VerCors* [13], the authors Raúl E. Monti, Robert Rubbens, and Marieke Huisman present their experience with verifying memory safety and data race freedom in an industrial case study of a concurrent module of a tunnel control system written in Java. The employed tool is VerCors, a software verification tool that requires annotating the code before carrying out verification. Using VerCors led to discovering two concurrency bugs related to memory access. The findings, as well as the underlying verification paradigm have been presented to the industrial partner, and feedback has been collected. Based on this feedback, the authors gain insight that could guide further improvement of VerCors and the entire verification process also.

In *Exploring a Parallel SCC Algorithm Using TLA+ and the TLC Model Checker* [18], the author Jaco van de Pol describes a case study on using the TLA+ model checker to increase the understanding of Bloemen's parallel SCC decomposition concurrent algorithm. The paper explains how the algorithm and correctness properties can be specified in TLA+ (both a naive version and an improved version), and reports on findings. For ten concrete instances (two workers on ten small graphs), the improved version of the specification satisfies the correctness properties. The paper also explains that various further modifications of the improved specification violate the correctness properties, sometimes unexpectedly.

In *A Formal Model of Metacontrol in Maude* [15], authors Juliane Päßler, Esther Aguado, Gustavo Rezende Silva, Silvia Lizeth Tapia Tarifa, Carlos Hernández Corbato, and Einar Broch Johnsen present an executable formal model of a self-adaptive system based on metacontrol, by formalizing a smart house-heating application in Maude. All the layers of the self-adaptive system of systems have been captured, and the evaluation for different scenarios of environment inputs and system failures has shown an improved performance of the smart house system with respect to its requirements when the Metacontroller performs architectural adaptation, when compared to the situation when the system's behavior is limited to one of the individual controllers.

## References

1. Blatter, L., Kosmatov, N., Prevosto, V., Le Gall, P.: An efficient vcgen-based modular verification of relational properties. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 498–516. Springer, Cham (2022)
2. Blom, S., Darabi, S., Huisman, M., Oortwijn, W.: The VerCors tool set: verification of parallel and concurrent software. In: Polikarpova, N., Schneider, S. (eds.) IFM 2017. The VerCors Tool Set: Verification of Parallel and Concurrent Software, vol. 10510, pp. 102–110. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66845-1_7
3. Bussi, L., Ciancia, V., Gadducci, F., Latella, D., Massink, M.: On binding in the spatial logic for closure spaces. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 479–497. Springer, Cham (2022)
4. Cai, S., Gallina, B., Nyström, D., Seceleanu, C.: Effective test suite design for detecting concurrency control faults in distributed transaction systems. In: Margaria, T., Steffen, B. (eds.) ISoLA 2018. LNCS, vol. 11246, pp. 355–374. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03424-5_24
5. da Rocha Pinto, P., Dinsdale-Young, T., Gardner, P.: TaDA: a logic for time and data abstraction. In: Jones, R. (ed.) ECOOP 2014. LNCS, vol. 8586, pp. 207–231. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44202-9_9
6. de Hoz Diego, J.D., Temperekidis, A., Katsaros, P., Konstantinou, C.: An iot digital twin for cyber-security defence based on runtime verification. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 556–574. Springer, Cham (2022)
7. Groote, J.F., Laveaux, M., van Spaendonck, P.H.M.: A thread-safe term library. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 422–459. Springer, Cham (2022)

8. Hawblitzel, C., et al.: Ironfleet: proving practical distributed systems correct. In: Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, pp. 1–17. ACM (2015)
9. Jacobs, B., Smans, J., Philippaerts, P., Vogels, F., Penninckx, W., Piessens. , F.: VeriFast: a powerful, sound, predictable, fast verifier for C and Java. In: NFM (2011)
10. Jongmans, S.-S., Proença, J.: St4mp: a blueprint of multiparty session typing for multilingual programming. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 460–478. Springer, Cham (2022)
11. Jung, R., et al.: Iris: monoids and invariants as an orthogonal basis for concurrent reasoning. In: POPL, pp. 637–650. ACM (2015)
12. Krogh-Jespersen, M., Timany, A., Ohlenbusch, M.E., Gregersen, S.O., Birkedal, L.: Aneris: a mechanised logic for modular reasoning about distributed systems. In: ESOP 2020. LNCS, vol. 12075, pp. 336–365. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44914-8_13
13. Monti, R.E., Rubbens, R., Huisman, M.: On deductive verification of an industrial concurrent software component with vercors. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 517–534. Springer, Cham (2022)
14. Müller, P., Schwerhoff, M., Summers, A.J.: Viper - a verification infrastructure for permission-based reasoning. In: VMCAI (2016)
15. Päßler, J., Aguado, E., Silva, G.R., Tarifa, S.L.T., Hernández Corbato, C., Johnsen, E.B.: A formal model of metacontrol in maude. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 575–596. Springer, Cham (2022)
16. Sergey, I., Nanevski, A., Banerjee, A.: Mechanized verification of fine-grained concurrent programs. In: PLDI, pp. 77–87. ACM (2015)
17. Sharma, R., Bauer, M., Aiken, A.: Verification of producer-consumer synchronization in GPU programs. In: Proceedings of PLDI 2015, pp. 88–98. ACM (2015)
18. van de Pol, J.: Exploring a parallel scc algorithm using tla+ and the tlc model checker. In: Margaria, T., Steffen, B. (eds.) ISoLA 2022. LNCS, vol. 13701, pp. 535–555. Springer, Cham (2022)