Studies in Computational Intelligence

Volume 817

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

The books of this series are submitted to indexing to Web of Science, EI-Compendex, DBLP, SCOPUS, Google Scholar and Springerlink.

More information about this series at http://www.springer.com/series/7092

Wiktor B. Daszczuk

Integrated Model of Distributed Systems

Springer

Wiktor B. Daszczuk
Institute of Computer Science
Warsaw University of Technology
Warsaw, Poland

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-030-12834-0 ISBN 978-3-030-12835-7 (eBook)
https://doi.org/10.1007/978-3-030-12835-7

Library of Congress Control Number: 2019930961

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

I dedicate this monograph to my friend Jurek Mieścicki

pamięci Jurka	in memory of Jurek
gdzie wędrował?	where he wandered?
trawa lekko skoszona jego stopami	grass mowed slightly by his feet
a gwiazdy tłumnie umierały	and the stars were dying in droves
robiąc miejsce dla zmyślonego smoka	making room for imaginary dragon
połykającego swój ogon	swallowing its tail
połykającego świat cały	swallowing the whole world
mrówki jego myśli toczyły wieczną bitwę	ants of his thoughts took an eternal battle
nacierając pod sztandarem mądrości	assaulting under the banner of wisdom
który padał zadeptany następną myślą	which felt trampled by a next thought
a sen o pajęczynie pamięci	and a dream of a spider's web of memory
uświadamiał spisek przyzwyczajeń	made someone conscious of conspiracy habits
podróżując aleją	traveling an alley
wzdłuż żywopłotu sekund	along the hedge of seconds
wyciągał rękę	he extended his hand
a małe zdarzenia	and small events
łaskotały	were tickling
i raniły	and hurting
przykuwały codziennością konieczności	attracted an everyday necessity
koniecznością codzienności	a necessity of everyday life
chciał tylko minąć następny kamień	he just wanted to get passed the next stone
lecz ten nie wydawał się inny od poprzednich	but this did not seem different from previous ones
ani bardziej szczęśliwy	nor more happy
ani bardziej mokry	nor wetter
ani bardziej egipski	nor more Egyptian
szczęśliwa gwiazda spadła	a lucky star felt down
robiąc wiele spustoszenia	doing a lot of havoc
zaśmiała się niecierpliwie	it laughed impatiently
na pogrzebie jutra	at the funeral of the next day

Preface

Deadlock, a situation in which the system or part of it just stops and cannot continue successfully, is a typical error identified in distributed computer systems. However, verification of computer systems often requires some knowledge about the mathematical background and formal methods from the designer. In many institutions, formal verification is necessarily required, for example, in avionics or NASA projects. On the other hand, many engineers and students give up formal verification because time and effort must be sacrificed to study formalism. In many cases, they only decide to test, without formal proofs of proper cooperation in distributed systems.

The author offers original formalism—Integrated Model of Distributed Systems (IMDS)—combined with model checking based on temporal logic. The formalism allows the designer a specification that emphasizes the natural features of distributed systems and provides automatic verification of the typical features of such systems, such as deadlock and termination. The verified system is defined in a uniform model that can be observed in two general perspectives highlighting different aspects of the system. The server view involves servers' states and message exchange. The agent view presents distributed agents traveling by means of messages and collaborating through shared resources.

The introduced IMDS formalism exploits natural features of distributed systems:

- *Locality*: Individual actions that are part of the processes are executed based on the local server's state and the set of currently pending messages at this server, regardless of what is happening in other servers. No incident from outside the server—except for messages sent to it—can influence the operation of the server. Servers have no access to any global or non-local variables.
- *Autonomy*: Every decision about executing an action on the server—including the choice between many possible actions—is made autonomously. The only manner to influence the behavior of the server is to send a message to it, which may enable a previously impossible action.

- *Asynchrony*: Any required synchronization is hidden inside processes; therefore, processes are perceived as asynchronous. In addition, unidirectional communication channels between servers are naturally observed as asynchronous. Ergo, asynchrony is built into the grounds of IMDS.

Many modeling formalisms are based on synchronous actions performed by processes or on synchronous communication. In the author's opinion, asynchronous behavior is more realistic in the description of distributed systems, in which the only way to contact another server is to send a message and wait for response. The IMDS formalism emphasizes the enumerated characteristics of distributed systems: locality, autonomy and asynchrony.

The main research contribution of this monograph is the formulation of IMDS, in which communication duality is exposed, and which supports locality, autonomy and asynchrony in the specification of distributed systems.

Integration of IMDS with model checking allows detecting different types of deadlock and checking distributed termination using general formulas. These general formulas are not related to any specific structure of the model being verified. Communication deadlocks are observed in the server view, while resource deadlocks are observed in the agent view. This dichotomy is a rarity among commonly used formalisms. An important feature of IMDS is also the automatic distinction between deadlocks and distributed termination, in which processes stop deliberately.

General formulas for deadlock detection and termination checking enable automatic verification of these features. Many formalisms include automatic verification of total features, which involve all processes in the system. Partial deadlocks and partial termination—involving only a subset of all processes—must be individually determined by the designer in terms of features of the system under check. This requires the designer's knowledge, time and effort. Some formalisms support the design process by automatic detection of partial deadlocks and/or termination, but at the expense of limiting themselves to a certain class of verified systems, for example, only cycling systems. This monograph analyzes some of such approaches with their limitations. IMDS in conjunction with model checking is free from such restrictions, because a large class of systems can be automatically checked for partial and total deadlocks and termination. The obvious limitations of the introduced formalism are the finiteness of the model (for the purpose of static model checking) and the lack of broadcast communication (because messages are process carriers).

In automatic verification, temporal formulas specifying the features are generated by the Dedan program, developed by the author. The user does not need to know any temporal logic. The Dedan tool reports an error and provides a counterexample leading to a deadlock or a non-terminating loop. In this way, the designer can work in the style of "specify and push the button." The integration of IMDS with model checking and elaboration of general temporal formulas to verify deadlocks and termination, including their partial form, are both the research and the practical contribution of the author.

For the presented IMDS formalism, the textual specification language is developed and a graphical form of distributed automata is introduced. In addition, the conversion to equivalent Petri nets is defined. These three equivalent specification methods are used for various kinds of analysis:

- basic textual form for model checking and simulation over global reachability space,
- distributed automata for simulation over individual distributed components,
- Petri net for finding structural properties of the verified system, such as separation of components or unreachable actions; in Petri net analysis, many deadlocks may be found using single verification, while model checking finds only one of possibly many deadlocks.

The Dedan program, developed by the author, is based on the introduced methodology of specification in IMDS and on temporal verification. The program supports the specification of distributed systems, in textual or graphical form. A conversion from the server view of the system to the agent view is done algorithmically. Temporal verification is hidden inside the program; the user sees the results in an easily readable form of sequence-diagram-like counterexamples. The internal TempoRG verifier and external, commonly used temporal verifiers are used for model checking. In addition, a global reachability space of a checked system may be explored manually and its behavior may be observed in several simulation modes. Conversions of system specification to Petri net or to distributed automata model are supported. The translation of IMDS into Petri net and to distributed automata DA^3, invented by the author, and the development of the Dedan program are the practical contribution.

External verifiers used in Dedan do not support some kinds of fairness, which sometimes results in reporting false deadlocks. The author's original verification algorithm checking by spheres (CBS) is free from the mentioned disadvantage. The algorithm was improved for Dedan program using reverse reachability, which supports identification of individual cases in the reachability space.

The imposition of time constraints on the actions executed in the verified system, and on the message passing channels, may change the behavior of the system. For example, some deadlocks may disappear, while other deadlocks may arise. The timed version of IMDS formalism is presented (T-IMDS), with conversion rules to timed automata, in order to verify distributed systems with real-time constraints. This conversion preserves the mentioned features of distributed systems: locality, autonomy and asynchrony.

In order to verify very large systems whose reachability spaces cannot be built even under external, efficient verifiers, the non-exhaustive search algorithm "2-vagabonds" was developed. It is unique in partial deadlock and partial termination checking. The first vagabond puts hypotheses about deadlock/lack of termination, while the second vagabond verifies these hypotheses. The algorithm has a very limited set of parameters, in contrast to typical non-exhaustive algorithms.

The strong-fairness-aware algorithm, the timed model of distributed systems, preserving communication duality, locality, autonomy and asynchrony, and the non-exhaustive search algorithm for partial deadlock and termination checking are research and practical contributions.

Warsaw, Poland Wiktor B. Daszczuk

Acknowledgements

Stanisław Chrobot, Włodzimierz Zuberek and Jerzy Mieścicki helped to formulate the current version of Integrated Model of Distributed Systems. Janusz Sosnowski, Piotr Parewicz, Andrzej Pająk and Bogdan Czejdo gave important advice.

The following fragments are taken from the following publications (with major or minor changes):

- Chapter 2: (Daszczuk 2017) permission by Oxford University Press, Great Clarendon Street, Oxford, UK, OX2 6DP,
- Chapters 3 and 4, Figures in Chap. 1: (Daszczuk 2018) CC BY 4.0 license by MDPI, St. Alban-Anlage 66, 4052 Basel, Switzerland,
- Figure 5.9: (Czejdo et al. 2016) CC BY 4.0 license by Instutyt Wydawnicy Spatium Sp. z o.o., 25 Czerwca lok. 62, 26–600 Radom, Poland,
- Section 6.5: (Daszczuk 2019), permission license by Springer Nature,
- Figure 7.1, Sects. 7.2 and 7.3: (Daszczuk and Zuberek 2018), permission license by Springer Nature,
- Chapter 9: (Daszczuk and Zuberek 2018), permission license by Springer Nature,
- Figure 12.1—permission by Maria Daszczuk.

References

Czejdo, B., Bhattacharya, S. … Daszczuk, W. B. (2016). Improving resilience of autonomous moving platforms by real-time analysis of their cooperation. *Autobusy-TEST*, *17*(6), 1294–1301. http://www.autobusy-test.com.pl/images/stories/Do_pobrania/2016/nr%206/logistyka/10_l_czejdo_bhattacharya_baszun_daszczuk.pdf

Daszczuk, W. B. (2017). Communication and resource deadlock analysis using IMDS formalism and model checking. *The Computer Journal*, *60*(5), 729–750. https://doi.org/10.1093/comjnl/bxw099

Daszczuk, W. B. (2018). Specification and verification in integrated model of distributed systems (IMDS). *MDPI Computers*, *7*(4), 1–26. https://doi.org/10.3390/computers7040065

Daszczuk, W. B. (2019). Asynchronous specification of production cell benchmark in integrated model of distributed systems. In R. Bembenik, L. Skonieczny, G. Protaziuk, M. Kryszkiewicz, & H. Rybinski (Eds.), *23rd International Symposium on Methodologies for Intelligent Systems, ISMIS 2017, Warsaw, Poland, 26–29 June 2017, Studies in Big Data, vol. 40* (pp. 115–129). Cham, Switzerland: Springer International Publishing. https://doi.org/10.1007/978-3-319-77604-0_9

Daszczuk, W. B., Zuberek, W. M. (2018). Deadlock detection in distributed systems using the IMDS formalism and petri nets. In W. Zamojski, J. Mazurkiewicz, J. Sugier, T. Walkowiak, & J. Kacprzyk (Eds.), *12th International Conference on Dependability and Complex Systems, DepCoS-RELCOMEX 2017, Brunów, Poland, 2–6 July 2017. AISC vol. 582* (pp. 118–130). Cham, Switzerland: Springer International Publishing. https://doi.org/10.1007/978-3-319-59415-6_12

Highlights

- IMDS formalism for specification of distributed systems, based on servers with their states and agents with their messages. Behavior modeled as actions over states and messages.
- Communication duality: The IMDS system decomposed into server processes communicating by messages or into agent processes communicating by servers' states. Two views of a system: server view and agent view.
- IMDS exposes natural features of distributed systems: locality of actions on servers, no global variables, autonomy of servers and asynchrony of processes and of communication.
- IMDS combined with model checking provides automatic verification of deadlocks: in communication or over resources, total and partial, and of distributed termination: total and partial.
- The Dedan program developed for specification, verification and simulation of distributed systems, based on IMDS formalism and model checking.
- Alternative formulation of IMDS in Petri nets for structural analysis: finding separate subsystems, unreachable actions and multiple deadlocks.
- Distributed automata DA^3, equivalent to IMDS—simulation over distributed components, preserved communication duality (two kinds of automata: server-based and agent-based).
- CBS, original verification algorithm, supporting strong fairness (compassion).
- Cooperation of Dedan with external model checkers: Spin, NuSMV, Timeless Uppaal.
- Real-time-constrained version of IMDS (T-IMDS) cooperating with external timed verifier Uppaal, translation of T-IMDS to Uppaal Timed Automata.
- 2-vagabonds: non-exhaustive verification algorithm for partial deadlock detection and partial termination checking.

Contents