

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

© Springer Nature Switzerland AG 2020

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, express or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

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

gdzie wędrował?
trawa lekko skoszona jego stopami
a gwiazdy tłumnie umierały
robiąc miejsce dla zmyślonego smoka
połykającego swój ogon
połykającego świat cały

mrowki jego myśli toczyły wieczną bitwę
nacierając pod sztandarem mądrości
który padał zdeptany następną myślą
a sen o pajęczynie pamięci
uświadamiał spisek przyzwyczajęń

podróżując aleją
wzdłuż żywopłotu sekund
wyciągał rękę
a małe zdarzenia
łaskotały
i raniły
przykuwały codziennością konieczności
koniecznością codzienności

chciał tylko minąć następny kamień
lecz ten nie wydawał się inny od poprzednich
ani bardziej szczęśliwy
ani bardziej mokry
ani bardziej egipski

szczęśliwa gwiazda spadła
robiąc wiele spustoszenia
zaśmiała się niecierpliwie
na pogrzebie jutra

in memory of Jurek

where he wandered?
grass mowed slightly by his feet
and the stars were dying in droves
making room for imaginary dragon
swallowing its tail
swallowing the whole world

ants of his thoughts took an eternal battle
assaulting under the banner of wisdom
which felt trampled by a next thought
and a dream of a spider's web of memory
made someone conscious of conspiracy habits

traveling an alley
along the hedge of seconds
he extended his hand
and small events
were tickling
and hurting
attracted an everyday necessity
a necessity of everyday life

he just wanted to get passed the next stone
but this did not seem different from previous ones
nor more happy
nor wetter
nor more Egyptian

a lucky star felt down
doing a lot of havoc
it laughed impatiently
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 DA³, 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 Instytut Wydawniczy 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_1_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

| | | |
|----------|---|----|
| 1 | Introduction | 1 |
| 1.1 | Overview of the Formalism | 1 |
| 1.2 | Modeling of Distributed Systems with Communication Duality | 7 |
| 1.3 | Distributed Deadlock and Termination | 9 |
| 1.4 | Practical Outcome | 11 |
| 1.5 | The Organization of the Monograph | 12 |
| | References | 12 |
| 2 | Related Work on Deadlock and Termination Detection Techniques | 17 |
| | References | 24 |
| 3 | Integrated Model of Distributed Systems | 31 |
| 3.1 | Basic IMDS Definition | 31 |
| 3.2 | IMDS System Behavior | 32 |
| 3.3 | IMDS Processes | 33 |
| 3.4 | Views of a Distributed System | 38 |
| 3.5 | IMDS as a Programming Language | 40 |
| 3.6 | Semantics | 44 |
| 3.7 | Deadlock and Termination | 47 |
| | References | 48 |
| 4 | Model Checking of IMDS Specifications in the Dedan Environment | 49 |
| | References | 51 |
| 5 | Deadlock Detection Examples: The Dedan Environment at Work | 53 |
| 5.1 | “Two Semaphores” Model | 53 |
| 5.2 | Dijkstra’s Philosophers | 60 |

- 5.3 Intersection 66
- 5.4 Automatic Vehicle Guidance System 69
- 5.5 Resource Deadlock Without Communication Deadlock 70
- 5.6 Production Cell 71
 - 5.6.1 Centralized Versus Distributed Modeling 73
 - 5.6.2 Synchrony Versus Asynchrony of Specification 73
 - 5.6.3 Real-Time Modeling 74
 - 5.6.4 Message Passing/Resource Sharing 74
 - 5.6.5 Synthesis and Verification 74
 - 5.6.6 Modeling Production Cell in IMDS 75
 - 5.6.7 Verification in Dedan 76
 - 5.6.8 Increasing Parallelism 78
- 5.7 Time of Verification 79
- 5.8 Use in Teaching—The Rybu Preprocessor 80
- References 82
- 6 Using the Dedan Program 87**
 - 6.1 Program Structure 87
 - 6.2 Main Window 88
 - 6.3 Files 90
 - 6.4 Model Name 90
 - 6.5 Server Type Definition 90
 - 6.6 Agent Type Definition 92
 - 6.7 Instances Declaration 92
 - 6.8 Initialization Part 92
 - 6.9 Macrogenerator 93
 - 6.10 Verification 93
 - 6.11 Agent View 96
 - References 97
- 7 Deadlock Detection in Petri Net Equivalent to IMDS 99**
 - 7.1 Petri Net Equivalent to IMDS 99
 - 7.1.1 Petri Net Definition 100
 - 7.1.2 Translation of IMDS to PN 101
 - 7.1.3 Example 102
 - 7.2 Deadlock Detection Using Siphons 105
 - 7.3 Problem with Not Purely Cyclic Systems 108
 - 7.4 Translation from Petri Net to IMDS 111
 - 7.4.1 Servers 111
 - 7.4.2 Agents 113
 - 7.4.3 Example 115
 - 7.4.4 Termination 115
 - 7.4.5 Processes 115
 - 7.4.6 Formal Requirements for Petri Net 118

- 7.4.7 Communication Duality, Locality, Autonomy
and Asynchrony 119
- 7.4.8 Semantics 122
- References 123
- 8 Distributed Autonomous and Asynchronous Automata (DA³) 125**
 - 8.1 Server Automata S-DA³ 126
 - 8.2 Agent Automata A-DA³ 130
 - 8.3 Using DA³ in the Dedan Program 133
 - References 136
- 9 Fairness in Distributed Systems Verification 139**
 - 9.1 The Benchmark—Bounded Buffer 140
 - 9.2 LTL Model Checking in Spin 143
 - 9.3 CTL and LTL Model Checking in NuSMV 145
 - 9.4 CTL Model Checking in Uppaal 150
 - 9.5 Fair Verification Algorithm 155
 - References 158
- 10 Timed IMDS 161**
 - 10.1 Timed Automata 161
 - 10.1.1 Uppaal TA Syntax 165
 - 10.1.2 UTA Semantics 166
 - 10.1.3 Network of UTA Syntax 166
 - 10.1.4 Network of UTA Semantics 167
 - 10.2 Timed Extensions to IMDS (T-IMDS) 168
 - 10.2.1 Asynchronous Channels with Limited Time
Delay (CT-IMDS) 169
 - 10.2.2 Time Bounds of States (BT-IMDS) 169
 - 10.3 Conversion of T-IMDS to Uppaal Timed Automata 170
 - 10.3.1 Implementation of BT-IMDS 172
 - 10.3.2 Implementation of CT-IMDS 175
 - 10.4 Formal Translation of CT-IMDS to Uppaal Timed
Automata 179
 - 10.4.1 Translation of Individual Servers to Uppaal Timed
Automata 180
 - 10.4.2 Semantics of Server Timed Automaton 182
 - 10.4.3 Translation of Asynchronous Channels to Uppaal
Timed Automaton 183
 - 10.4.4 Semantics of Asynchronous Channel Automaton 184
 - 10.4.5 Translation of CT-IMDS System to Uppaal Timed
Automata 184
 - 10.4.6 Semantics of CT-IMDS System Implemented
as Uppaal Timed Automata 186

- 10.5 Example of Timed IMDS System—Two Semaphores 187
- 10.6 Example of Timed IMDS System—Automatic Vehicle
Guidance System 190
- 10.7 Verification of the Production Cell 190
- References 191
- 11 2-Vagabonds: Non-exhaustive Verification Algorithm 193**
 - 11.1 Related Work 195
 - 11.2 2-Vagabonds Algorithm 196
 - 11.2.1 Total Deadlock 196
 - 11.2.2 Partial Server Deadlock (Communication Deadlock) . . . 198
 - 11.2.3 Partial Agent Deadlock (Resource Deadlock) 200
 - 11.2.4 Partial Distributed Termination 200
 - 11.2.5 Heuristics 201
 - 11.2.6 Verification 202
 - 11.3 Experimental Results 203
 - 11.3.1 The Benchmarks 203
 - 11.3.2 Results 204
 - 11.4 Parallel Verification 211
 - 11.5 Limitations 212
 - References 216
- 12 Conclusions and Further Work 219**
 - References 225
- Appendix A: Acronyms, Shortcuts and Symbols Used in the Text 227**
- Appendix B: IMDS Formal Definition 233**
- Appendix C: IMDS Syntax 235**