Lecture Notes in Computer Science 11194

More information about this series at http://www.springer.com/series/7407

Jules Desharnais · Walter Guttmann
Stef Joosten (Eds.)

Relational and Algebraic Methods in Computer Science

17th International Conference, RAMiCS 2018
Groningen, The Netherlands, October 29 – November 1, 2018
Proceedings

Springer

Editors
Jules Desharnais
Université Laval
Québec, QC
Canada

Stef Joosten
Open Universiteit
Heerlen
The Netherlands

Walter Guttmann
University of Canterbury
Christchurch
New Zealand

ISSN 0302-9743 ISSN 1611-3349 (electronic)
Lecture Notes in Computer Science
ISBN 978-3-030-02148-1 ISBN 978-3-030-02149-8 (eBook)
https://doi.org/10.1007/978-3-030-02149-8

Library of Congress Control Number: 2015949476

LNCS Sublibrary: SL1 – Theoretical Computer Science and General Issues

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

Preface

This volume contains the proceedings of the 17th International Conference on Relational and Algebraic Methods in Computer Science (RAMiCS 2018), which was held in Groningen, The Netherlands, from October 29 to November 1, 2018.

The plan to initiate this series of conferences was put in place during the 38th Banach Semester on Algebraic Methods in Logic and Their Computer Science Application in Warsaw, Poland, in September and October 1991. The first numbered occurrence was a Dagstuhl seminar on Relational Methods in Computer Science (RelMiCS 1), held in Germany in 1994. From then on and until 2009, at intervals of about one year and a half, there was a RelMiCS conference. Starting in 2003, RelMiCS conferences were held jointly with Applications of Kleene Algebras (AKA) conferences. At RelMiCS 11/AKA 6 in Doha, Qatar, it was decided to use the current name for the series, that is, Relational and Algebraic Methods in Computer Science (RAMiCS).

Recurrent topics of RAMiCS conferences include semiring- and lattice-based structures such as relation algebras and Kleene algebras, their connections with program logics and other logics, their use in theories of computing, their formalization with theorem provers, and their application to modeling and reasoning about computing systems and processes.

In total, 30 papers were submitted to RAMiCS 2018 and the Program Committee selected 21 of them for presentation at the conference. In these proceedings the selected papers are grouped under three headings: "Theoretical Foundations," "Reasoning About Computations and Programs," and "Applications and Tools." Each submission was evaluated by at least four independent reviewers, and further discussed electronically during two weeks. The chairs are very grateful to all Program Committee members and to the external reviewers for their time and expertise.

Roland Backhouse, Manuel Bodirsky, and Philippa Gardner kindly accepted our invitation to present their research at the conference. The abstracts of Roland Backhouse's talk, "The Importance of Factorisation in Algorithm Design," and Philippa Gardner's talk, "Scalable Reasoning About Concurrent Programs," are included in the proceedings. So is the full paper related to Manuel Bodirsky's talk, "Finite Relation Algebras with Normal Representations." This conference featured a tutorial on motivating students for relation algebra presented by one of us (Stef Joosten). The abstract of this tutorial is also included in this volume. An experimental session of this conference was a meeting with software engineers who work in practice, most of whom have little contact with the theories discussed in our conference. This meeting was planned as a way to foster applications of relational and algebraic methods in computer science.

We thank the RAMiCS Steering Committee for their support and the Open Universiteit Nederland for organizing this conference. We gratefully acknowledge the financial support of Ordina Nederland BV for their sponsorship and their help with organizing this

conference. We also thank the Groningen Congres Bureau and the Rijksuniversiteit Groningen for the warm welcome we received in the Academiegebouw and the city itself. A special thank you is due to Sebastiaan J. C. Joosten, author in previous editions and this time a very active, talented publicity chair.

We also appreciate the excellent facilities offered by the EasyChair conference administration system, and Alfred Hofmann and Anna Kramer's help in publishing this volume with Springer. Finally, we owe much to all authors and participants for their support of this RAMiCS conference.

October 2018

Jules Desharnais
Walter Guttmann
Stef Joosten

Organization

Organizing Committee

Conference Chair

Stef Joosten	Open Universiteit, The Netherlands

PC Chairs

Jules Desharnais	Université Laval, Canada
Walter Guttmann	University of Canterbury, New Zealand

Publicity Chair

Sebastiaan Joosten	Universiteit Twente, The Netherlands

Program Committee

Luca Aceto	Reykjavík University, Iceland Gran Sasso Science Institute, Italy
Rudolf Berghammer	Christian-Albrechts-Universität zu Kiel, Germany
Jules Desharnais	Université Laval, Canada
Uli Fahrenberg	École Polytechnique, France
Hitoshi Furusawa	Kagoshima University, Japan
Walter Guttmann	University of Canterbury, New Zealand
Robin Hirsch	University College London, UK
Peter Höfner	Data61, CSIRO, Australia
Marcel Jackson	La Trobe University, Australia
Jean-Baptiste Jeannin	University of Michigan, USA
Peter Jipsen	Chapman University, USA
Stef Joosten	Open Universiteit, The Netherlands
Wolfram Kahl	McMaster University, Canada
Barbara König	Universität Duisburg-Essen, Germany
Dexter Kozen	Cornell University, USA
Agi Kurucz	King's College London, UK
Tadeusz Litak	Friedrich-Alexander Universität Erlangen-Nürnberg, Germany
Roger Maddux	Iowa State University, USA
Annabelle McIver	Macquarie University, Australia
Szabolcs Mikulás	Birkbeck, University of London, UK
Ali Mili	New Jersey Institute of Technology, USA
Bernhard Möller	Universität Augsburg, Germany
José N. Oliveira	Universidade do Minho, Portugal
Alessandra Palmigiano	Technische Universiteit Delft, The Netherlands

Damien Pous	CNRS, France
Mehrnoosh Sadrzadeh	Queen Mary University of London, UK
John Stell	University of Leeds, UK
Georg Struth	University of Sheffield, UK
Michael Winter	Brock University, Canada

Steering Committee

Rudolf Berghammer	Christian-Albrechts-Universität zu Kiel, Germany
Jules Desharnais	Université Laval, Canada
Ali Jaoua	Qatar University, Qatar
Peter Jipsen	Chapman University, USA
Bernhard Möller	Universität Augsburg, Germany
José N. Oliveira	Universidade do Minho, Portugal
Ewa Orłowska	National Institute of Telecommunications, Poland
Gunther Schmidt	Universität der Bundeswehr München, Germany
Michael Winter (Chair)	Brock University, Canada

Additional Reviewers

Musa Al-Hassy
Benjamin Cabrera
Christian Doczkal
Jérémy Dubut
Sabine Frittella
Ilias Garnier
Roland Glück
Giuseppe Greco
Lucy Ham
Sebastiaan Joosten
Fei Liang
Konstantinos Mamouras
Koki Nishizawa
Filip Sieczkowski
Jorge Sousa Pinto
Norihiro Tsumagari
Thorsten Wißmann

Sponsors

Ordina Nederland BV (Main Sponsor)
Groningen Congres Bureau

Local Organizers

Chrisja Muris	Open Universiteit
Heleen Bakker	Open Universiteit
Hillebrand Meijer	Ordina
Erick Koster	Ordina
Astrid Patz	Ordina
Margot Spee	Groningen Congres Bureau
Kelly Scholtens	Groningen Congres Bureau

Abstracts of Invited Talks

Scalable Reasoning About Concurrent Programs

Philippa Gardner

Imperial College London, UK
p.gardner@imperial.ac.uk

Scalable reasoning about complex concurrent programs interacting with shared memory is a fundamental, open research problem. Developers manage the complexity of concurrent software systems by designing software components that are *compositional* and *modular*. With compositionality, a developer designs local subcomponents with well-understood *interfaces* that connect to the rest of the system. With modularity, a developer designs reusable subcomponents with *abstract* software interfaces that can hide the complexity of the subcomponents from the rest of the system. The challenge is to develop compositional, modular reasoning of concurrent programs, which follows the intuitions of the developer in how to structure their software components with precisely defined specifications of software interfaces. These specifications should not leak implementation details and should be expressed at the level of abstraction of the client.

I will describe the work done by my group and others on compositional and modular reasoning about concurrent programs using modern concurrent separation logics. I will present work on reasoning about *safety properties*, highlighting the CAP logic [ECOOP'10] which introduced logical abstraction (the fiction of separation) to concurrent separation logics and the TaDA logic [ECOOP'14] which introduced abstract atomicity (the fiction of atomicity). I will also present new work on *progress properties*, introducing the TaDA-Live logic for reasoning about the termination of blocking programs. I will demonstrate the subtlety of the reasoning using a simple lock module, and also compare this work with linearizability, contextual refinement and other concurrent separation logics.

Papers to read:

O'Hearn, P.W.: Resources, concurrency, and local reasoning. Theor. Comput. Sci. **375** (1–3), 271–307 (2007)
http://www0.cs.ucl.ac.uk/staff/p.ohearn/papers/concurrency.pdf
Dinsdale-Young, T., da Rocha Pinto, P., Gardner, P.: A perspective on specifying and verifying concurrent modules. J. Logical Algebraic Methods Program. **98**, 1–25 (2018)
https://www.doc.ic.ac.uk/~pg/publications/Dinsdale-Young2018perspective.pdf

The Importance of Factorisation in Algorithm Design

Roland Backhouse

School of Computer Science, University of Nottingham
rcb@cs.nott.ac.uk

In 1971, J.H. Conway [Con71] published a slim volume entitled "Regular Algebra and Finite Machines" which was to have great influence on my own work (eg. [Bac06]). I was particularly impressed by the chapter on factor theory and its subsequent application in the construction of biregulators. Although some elements of Conway's book are now well cited, this part of the book still appears to be much less well known. The goal of this talk is to explain why factor theory is important in the design of algorithms.

We introduce Conway's factor matrix and then show how the (unique) reflexive-transitive-reduction of the factor matrix, dubbed the "factor graph" [Bac75], is the basis of the well-known Knuth-Morris-Pratt pattern-matching algorithm [KMP77, BL77]. This serves as an appetiser for a review of fixed-point theory and Galois connections, focusing particularly on the relevance of the theory in the design of algorithms.

We then return to factor theory and how it forms the basis of practical applications in program analysis [SdML04]. We conclude with some speculation on how a greater focus on factorisation might help us to better understand the complexity of algorithms.

References

[Bac75] Backhouse, R.C.: Closure algorithms and the star-height problem of regular languages. Ph.D. thesis, University of London (1975). Scanned-in copy of the chapters on factor theory available from www.cs.nott.ac.uk/~psarb2/MPC/FactorGraphs.pdf

[Bac06] Backhouse, R.: Regular algebra applied to language problems. J. Log. Algebraic Program. **66**, 71–111 (2006)

[BL77] Backhouse, R.C., Lutz, R.K.: Factor graphs, failure functions and bi-trees. In: Salomaa, A., Steinby, M. (eds.) ICALP 1977. LNCS, vol. 52, pp. 61–75. Springer, Heidelberg (1977)

[Con71] Conway, J.H.: Regular Algebra and Finite Machines. Chapman and Hall, London (1971)

[KMP77] Knuth, D.E., Morris, J.H., Pratt, V.R.: Fast pattern matching in strings. SIAM J. Comput. **6**, 325–350 (1977)

[SdML04] Sittampalam, G., de Moor, O., Larssen, K.F.: Incremental execution of transformation specifications. In: Proceedings of the 31st SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2004, vol. 39. ACM SIGPLAN Notices, pp. 26–38, January 2004

Tutorial: Relation Algebra in the Classroom with Ampersand

Stef Joosten

Open University of the Netherlands
stef.joosten@ou.nl

This tutorial explores a way to motivate students for relation algebras by applying it in software engineering. Participants will get hands-on experience with Ampersand [1], which is a compiler that transforms a specification in relation algebra in working software. They will be directed to the documentation site [2] for more precise information about the language and tools.

This tutorial starts with a motivation: Why might working with Ampersand motivate students for relation algebra? Then the participants will create an information system online, just like students do in the course Rule Based Design [3]. For this purpose we ask participants to bring their laptops. The presentation proceeds by pointing out which learning points are relevant in the tutorial. It finalizes by giving an overview in the available materials and tools. All materials are freely available in open source, so participants can take it to their own classrooms.

Professors who want to use these materials are cordially invited to partake in the further development.

Background

Ampersand was originally intended as a means to specify requirements [4] in heterogeneous relation algebra [5]. The toolset evolved into a tool for students [6]. The Ampersand toolset has been used since 2013 [7] at the Open University in two courses and numerous research assignments.

The novel feature of Ampersand is that a theory in relation algebra is being used as a database program. It specifies persistence (i.e. the database) and user interfaces. This is achieved by using one interpretation of the algebra: a relation is interpreted as a finite set of pairs.

The user gets a programming language that is declarative, strongly typed, and easily subjected to proofs of correctness [8]. The benefits are fast development (because the Ampersand compiler generates working software), maintainability (because software is easily divided into independent chunks), and adaptability (because generating an application and deploying it is automated).

References

1. Joosten, S.: RAP3 (2009–2018). rap.cs.ou.nl/RAP3
2. Joosten, S., Hageraats, E.: Documentation of Ampersand (2016–2018). ampersandtarski.gitbook.io/documentation/
3. Rutledge, L., Wetering, R.v.d., Joosten, S.: Course IM0103 Rule Based Design for CS (2009–2018). www.ou.nl/-/IM0403_Rule-Based-Design
4. Dijkman, R.M., Ferreira Pires, L., Joosten, S.M.M.: Calculating with concepts: a technique for the development of business process support. In: Evans, A. (ed.) Proceedings of the UML 2001 Workshop on Practical UML-Based Rigorous Development Methods: Countering or Integrating the eXstremists, Lecture Notes in Informatics, vol. 7. FIZ, Karlsruhe (2001)
5. van der Woude, J., Joosten, S.: Relational heterogeneity relaxed by subtyping. In: de Swart, H. (ed.) RAMICS 2011. LNCS, vol. 6663, pp. 347–361. Springer, Heidelberg (2011)
6. Michels, G., Joosten, S., van der Woude, J., Joosten, S.: Ampersand: applying relation algebra in practice. In: Proceedings of the 12th Conference on Relational and Algebraic Methods in Computer Science RAMICS 2011. LNCS, vol. 6663, pp. 280–293. Springer-Verlag, Berlin (2011)
7. Michels, G., Joosten, S.: Progressive development and teaching with RAP. In: Proceedings of the 3rd Computer Science Education Research Conference on Computer Science Education Research, CSERC 2013, pp. 3:33–3:43. Open University, Heerlen, The Netherlands (2013)
8. Joosten, S.: Relation algebra as programming language using the ampersand compiler. J. Log. Algebraic Methods Program. **100**, 113–129 (2018)

Contents

Invited Paper

Reasoning About Computations and Programs

Applications and Tools