ICT COST Action IC1405

WG1 Year-End Report
COST Action IC1405 Reversible Computation

Editors: Iain Phillips and Michael Kirkedal Thomsen

May 2018

List of contributors

Gabriel Ciobanu
Robert Glück
Robin Kaarsgaard
Claudio Antares Mezzina
Łukasz Mikulski
Rajagopal Nagarajan
Luca Paolini
Irek Ulidowski
German Vidal
Thomas Worsch

Contents

1 Introduction 4

2 Finite-State Computing Models 4

3 Reversible Cellular Automata 4

4 Theory of Programming Languages 4

5 Model-based Testing 6

6 Term Rewriting 6

7 Categorical models and semantics 6

8 Petri Nets 7

9 Process Calculi 8

10 Membrane Computing 9

11 Formal Verification of Quantum Systems 10

1 Introduction

This report covers research carried out with the aid of COST Action IC1405 on Reversible Computation during the third grant period GP3 (May 2017 to April 2018); and in particular research relating to the topics covered by Working Group WG1 Foundations. We have mostly followed the structure of the State of the Art report for WG1, but we have added the topics of 'Theory of Programming Languages' and 'Membrane Computing.' Note that work on Programming Languages is also to be found in the Year-End Report of WG2 Software and Systems.

Iain Phillips and Michael Kirkedal Thomsen

2 Finite-State Computing Models

Work continues but there are no new publications to report for the third grant period GP3.

3 Reversible Cellular Automata

[KSW18] Jarkko Kari, Ville Salo, and Thomas Worsch. Sequentializing cellular automata. *CoRR*, abs/1802.06668, 2018

The work by Jarkko Kari, Ville Salo and Thomas Worsch on the application of reversible block rules for CA in an asynchronous setting has led to a first cornerstone.

We now have a precise characterization of the one-dimensional CA that can be realized by applying a reversible block rule sequentially in a "full sweep from left to right" (or right to left) at all positions. It turns out that not all reversible CA can be realized, but also some non-reversible ones. It is decidable whether a CA can be realized that way or not.

Thomas Worsch

4 Theory of Programming Languages

[PPR18] Luca Paolini, Mauro Piccolo, and Luca Roversi. A certified study of a reversible programming language. In *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18-21, 2015, Tallinn, Estonia*, volume 69 of *LIPIcs*, pages 7:1–7:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018

We advance in the study of the semantics of Janus, a C-like reversible programming language. Our study makes utterly explicit some backward and forward evaluation symmetries. We want to deepen mathematical knowledge about the foundations and design principles of reversible computing and programming languages. We formalize a big-step operational semantics and a denotational semantics of Janus. We show a full abstraction result between the operational and denotational semantics. Last, we certify our results by means of the proof assistant Matita.

> [PZ17] Luca Paolini and Margherita Zorzi. qpcf: A language for quantum circuit computations. In *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Bern, Switzerland, April 20-22, 2017, Proceedings*, volume 10185 of *Lecture Notes in Computer Science*, pages 455–469, 2017

We propose qPCF, a functional language able to define and manipulate quantum circuits in an easy and intuitive way. qPCF follows the tradition of "quantum data & classical control" languages, inspired to the QRAM model. Ideally, qPCF computes finite circuit descriptions which are offloaded to a quantum co-processor (i.e. a quantum device) for the execution. qPCF extends PCF with a new kind of datatype: quantum circuits. The typing of qPCF is quite different from the mainstream of "quantum data & classical control" languages that involves linear/exponential modalities. qPCF uses a simple form of dependent types to manage circuits and an implicit form of monad to manage quantum states via a destructive-measurement operator.

The following publication relates to GP2:

> [PPR16] Luca Paolini, Mauro Piccolo, and Luca Roversi. A class of reversible primitive recursive functions. *Electr. Notes Theor. Comput. Sci.*, 322:227–242, 2016

Reversible computing is bi-deterministic which means that its execution is both forward and backward deterministic, i.e. next/previous computational step is uniquely determined. Various approaches exist to catch its extensional or intensional aspects and properties. We present a class RPRF of reversible functions which holds at bay intensional aspects and emphasizes the extensional side of the reversible computation by following the style of Dedekind-Robinson Primitive Recursive Functions. The class RPRF is closed by inversion, can only express bijections on integers — not only natural numbers —, and it is expressive enough to simulate Primitive Recursive Functions, of course, in an effective way.

Luca Paolini

The following publication relates to GP2:

> [HMG17] Tue Haulund, Torben Ægidius Mogensen, and Robert Glück. Implementing reversible object-oriented language features on reversible machines. In *Reversible Computation - 9th International Conference, RC 2017, Kolkata, India, July 6-7, 2017, Proceedings*, volume 10301 of *Lecture Notes in Computer Science*, pages 66–73. Springer, 2017

We extended the reversible language Janus with support for class-based object-oriented programming, class inheritance and subtype-polymorphism. We describe how to implement these features on reversible hardware - with emphasis on the implementation of reversible dynamic dispatch using virtual method tables. Our translation is effective (i.e. garbage-free) and we demonstrate its practicality by implementation of a fully-featured compiler targeting the reversible assembly language PISA.

Robert Glück

5 Model-based Testing

Work continues but there are no new publications to report for the third grant period GP3.

6 Term Rewriting

[NPV17] Naoki Nishida, Adrián Palacios, and Germán Vidal. Reversible computation in term rewriting. *CoRR*, abs/1710.02804, 2017

Essentially, in a reversible programming language, for each forward computation from state S to state S', there exists a constructive method to go backwards from state S' to state S. Besides its theoretical interest, reversible computation is a fundamental concept which is relevant in many different areas like cellular automata, bidirectional program transformation, or quantum computing, to name a few. In this work, we focus on term rewriting, a computation model that underlies most rule-based programming languages. In general, term rewriting is not reversible, even for injective functions; namely, given a rewrite step $t_1 \to t_2$, we do not always have a decidable method to get t_1 from t_2. Here, we introduce a conservative extension of term rewriting that becomes reversible. Furthermore, we also define two transformations, injectivization and inversion, to make a rewrite system reversible using standard term rewriting. We illustrate the usefulness of our transformations in the context of bidirectional program transformation.

To appear in the Journal of Logical and Algebraic Methods in Programming.

German Vidal

7 Categorical models and semantics

[GK18] Robert Glück and Robin Kaarsgaard. A categorical foundation for structured reversible flowchart languages. *Electr. Notes Theor. Comput. Sci.*, 336:155–171, 2018

Structured reversible flowchart languages is a class of imperative reversible programming languages allowing for a simple diagrammatic representation of control flow built from a limited set of control flow structures, as ordinary structured flowcharts allow for conventional languages. This class includes the reversible programming language Janus (without recursion), as well as more recently developed reversible programming languages such as R-CORE and R-WHILE.

In the present paper, we develop a categorical foundation for this class of languages based on inverse categories with joins. We generalize the notion of extensivity of restriction categories to one that may be accommodated by inverse categories, and use the resulting decision maps to give a reversible representation of predicates and assertions. This leads to a categorical semantics for structured reversible flowcharts, from which we show that a program inverter can be extracted. Finally, we exemplify our approach by the development of a small structured reversible flowchart language, use our framework to both straightforwardly give it semantics and derive fundamental theorems about it, and discuss further applications of decisions in reversible programming.

Robin Kaarsgaard

8 Petri Nets

[dFEKM18] David de Frutos Escrig, Maciej Koutny, and Lukasz Mikulski. An efficient characterization of petri net solvable binary words. In *Applications and Theory of Petri Nets and Concurrency*, volume 10877 of *Lecture Notes in Computer Science*. Springer, 2018

The work by David de Frutos Escrig, Maciej Koutny and Lukasz Mikulski improved the characterization of binary Petri net solvable words and defined on their base reversible binary words.

We present a simple characterization of the set of Petri Net solvable binary words, that states that these are exactly the extensions of the prefixes of Petri Net cyclic solvable words, by some prefix x^k, where x can be any letter of considered binary alphabet, and k is any natural number. As a byproduct of the characterization, we also present a linear time algorithm to decide whether a word is solvable. The key idea is that the connection with the set of cyclic solvable words induces a regularity character, so that we just need to look for possible irregularities, and this can be done in a structural way, at the end producing the surprising linearity of the obtained decision algorithm. Finally, we utilize the discussed results in order to characterize (Petri net) reversible binary transition systems.

Łukasz Mikulski

9 Process Calculi

[MM16] Doriana Medic and Claudio Antares Mezzina. Static VS dynamic reversibility in CCS. In Simon J. Devitt and Ivan Lanese, editors, *Reversible Computation - 8th International Conference, RC 2016*, volume 9720 of *Lecture Notes in Computer Science*, pages 36–51. Springer, 2016

In the literature there exist two reversible variants of CCS. Reversible CCS (RCCS), proposed by Danos and Krivine, enacts reversibility by means of memory stacks. Ulidowski and Phillips proposed a general method to reverse a process calculus given in a particular SOS format, by exploiting the idea of making all the operators of a calculus static. CCSK is then derived from CCS with this method. Hence a natural question arises. Are these two reversible CCSs similar? In [MM16] a positive answer is given to this question, where an encoding from CCSK to RCCS and its opposite are presented.

[MK17] Claudio Antares Mezzina and Vasileios Koutavas. A safety and liveness theory for total reversibility. In Frédéric Mallet, Min Zhang, and Eric Madelaine, editors, *11th International Symposium on Theoretical Aspects of Software Engineering, TASE 2017, Sophia Antipolis, France, September 13-15, 2017*, pages 1–8. IEEE, 2017

We study the theory of safety and liveness in a reversible calculus where reductions are totally ordered and rollbacks lead systems to past states. Liveness and safety in this setting naturally correspond to the should-testing and inverse may-testing preorders, respectively. In reversible languages, however, the natural models of these preorders would need to be based on both forward and backward transitions, thus offering complex proof techniques for verification. Here we develop novel fully abstract models of liveness and safety which are based on forward transitions and limited rollback points, giving rise to considerably simpler proof techniques. Moreover, we show that, with respect to safety, total reversibility is a conservative extension to CCS. With respect to liveness, we prove that adding total reversibility to CCS distinguishes more systems. To our knowledge, this work provides the first testing theory for a reversible calculus, and paves the way for a testing theory for causal reversibility.

We have identified two sufficient properties for our safety and liveness theories to apply to any reversible language. In fact, the first property holds in languages with controlled, causal reversibility and therefore our safety theory immediately applies to them. The problem of determining whether the second property applies to remains open. We view the identification of this property as a first important step towards the development of a liveness theory for controlled, causal reversibility.

Claudio Antares Mezzina

[KU18] Stefan Kuhn and Irek Ulidowski. Local reversibility in a calculus of covalent bonding. *Sci. Comput. Program.*, 151:18–47, 2018

The paper introduces a process calculus with a new prefixing operator that allows us to model locally controlled reversibility. Actions can be undone spontaneously, as in other reversible process calculi, or as pairs of concerted actions, where performing a weak action forces undoing of another action. The new operator in its full generality allows us to model out-of-causal order computation, where effects are undone before their causes are undone, which goes beyond what typical reversible calculi can express. However, the core calculus, which uses only the reduced form of the new operator, is well behaved as it satisfied causal consistency. We demonstrate the usefulness of the calculus by modelling the hydration of formaldehyde in water into methanediol, an industrially important reaction, where the creation and breaking of some bonds are examples of locally controlled out-of-causal order computation.

Irek Ulidowski

10 Membrane Computing

[AC17] Bogdan Aman and Gabriel Ciobanu. Reversibility in parallel rewriting systems. *J. UCS*, 23(7):692–703, 2017

[AC18] Bogdan Aman and Gabriel Ciobanu. Controlled reversibility in reaction systems. In *Membrane Computing - 18th International Conference, CMC 2017, Bradford, UK, July 25-28, 2017, Revised Selected Papers*, volume 10725 of *Lecture Notes in Computer Science*, pages 40–53. Springer, 2018

Natural computing is a complex field of research dealing with models and computational techniques inspired by nature that helps us in understanding the biochemical world in terms of information processing. The articles [AC17] and [AC18] investigate the reversibility of biochemical reactions in two important theories of natural computing inspired by the functioning of living cells, namely in membrane computing and in reaction systems. Membrane computing deals with multisets of symbols processed in the compartments of a membrane structure according to some multiset rewriting rules; some of the symbols (presented with their multiplicity within the regions delimited by membranes) evolve in parallel according to the rules associated with their membranes, while the others remain unchanged and can be used in the subsequent steps. The situation is different in reaction systems; these systems represent a qualitative model, and so they deal with sets rather than multisets. Two major assumptions distinguish the reaction systems from the membrane systems:

(i) threshold assumption claiming that if a resource is present in the system, then it is present in a ”sufficient amount” such that several reactions needing such a resource are not in conflict (this means that reaction systems have actually an infinite multiplicity for their resources);

(ii) no permanency assumption claiming that an entity disappears from the current state unless it is produced by one of the reactions enabled in that state.

The important innovations of these articles are given by adding the reverse rules to the initial set of rules, as well as by adding an external control specified by using a special symbol informing the system that a rollback is needed. Their contribution is done by several theoretical results relating the evolutions of these systems to their reversible extensions.

Gabriel Ciobanu

11 Formal Verification of Quantum Systems

[PMNW17] Alessandra Di Pierro, Riccardo Mengoni, Rajagopal Nagarajan, and David Windridge. Hamming distance kernelisation via topological quantum computation. In *Theory and Practice of Natural Computing - 6th International Conference, TPNC 2017, Prague, Czech Republic, December 18-20, 2017, Proceedings*, volume 10687 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2017

We present a novel approach to computing Hamming distance and its kernelisation within Topological Quantum Computation. This approach is based on an encoding of two binary strings into a topological Hilbert space, whose inner product yields a natural Hamming distance kernel on the two strings. Kernelisation forges a link with the field of Machine Learning, particularly in relation to binary classifiers such as the Support Vector Machine (SVM). This makes our approach of potential interest to the quantum machine learning community.

Rajagopal Nagarajan

References

[AC17] Bogdan Aman and Gabriel Ciobanu. Reversibility in parallel rewriting systems. *J. UCS*, 23(7):692–703, 2017.

[AC18] Bogdan Aman and Gabriel Ciobanu. Controlled reversibility in reaction systems. In *Membrane Computing - 18th International Conference, CMC 2017, Bradford, UK, July 25-28, 2017, Revised Selected Papers*, volume 10725 of *Lecture Notes in Computer Science*, pages 40–53. Springer, 2018.

[dFEKM18] David de Frutos Escrig, Maciej Koutny, and Lukasz Mikulski. An efficient characterization of petri net solvable binary words. In *Applications and Theory of Petri Nets and Concurrency*, volume 10877 of *Lecture Notes in Computer Science*. Springer, 2018.

[GK18] Robert Glück and Robin Kaarsgaard. A categorical foundation for structured reversible flowchart languages. *Electr. Notes Theor. Comput. Sci.*, 336:155–171, 2018.

[HMG17] Tue Haulund, Torben Ægidius Mogensen, and Robert Glück. Implementing reversible object-oriented language features on reversible machines. In *Reversible Computation - 9th International Conference, RC 2017, Kolkata, India, July 6-7, 2017, Proceedings*, volume 10301 of *Lecture Notes in Computer Science*, pages 66–73. Springer, 2017.

[KSW18] Jarkko Kari, Ville Salo, and Thomas Worsch. Sequentializing cellular automata. *CoRR*, abs/1802.06668, 2018.

[KU18] Stefan Kuhn and Irek Ulidowski. Local reversibility in a calculus of covalent bonding. *Sci. Comput. Program.*, 151:18–47, 2018.

[MK17] Claudio Antares Mezzina and Vasileios Koutavas. A safety and liveness theory for total reversibility. In Frédéric Mallet, Min Zhang, and Eric Madelaine, editors, *11th International Symposium on Theoretical Aspects of Software Engineering, TASE 2017, Sophia Antipolis, France, September 13-15, 2017*, pages 1–8. IEEE, 2017.

[MM16] Doriana Medic and Claudio Antares Mezzina. Static VS dynamic reversibility in CCS. In Simon J. Devitt and Ivan Lanese, editors, *Reversible Computation - 8th International Conference, RC 2016*, volume 9720 of *Lecture Notes in Computer Science*, pages 36–51. Springer, 2016.

[NPV17] Naoki Nishida, Adrián Palacios, and Germán Vidal. Reversible computation in term rewriting. *CoRR*, abs/1710.02804, 2017.

[PMNW17] Alessandra Di Pierro, Riccardo Mengoni, Rajagopal Nagarajan, and David Windridge. Hamming distance kernelisation via topological quantum computation. In *Theory and Practice of Natural Computing - 6th International Conference, TPNC 2017, Prague, Czech Republic, December 18-20, 2017, Proceedings*, volume 10687 of *Lecture Notes in Computer Science*, pages 269–280. Springer, 2017.

[PPR16] Luca Paolini, Mauro Piccolo, and Luca Roversi. A class of reversible primitive recursive functions. *Electr. Notes Theor. Comput. Sci.*, 322:227–242, 2016.

[PPR18] Luca Paolini, Mauro Piccolo, and Luca Roversi. A certified study of a reversible programming language. In *21st International Conference on Types for Proofs and Programs, TYPES 2015, May 18-21, 2015, Tallinn, Estonia*, volume 69 of *LIPIcs*, pages 7:1–7:21. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[PZ17] Luca Paolini and Margherita Zorzi. qpcf: A language for quantum circuit computations. In *Theory and Applications of Models of Computation - 14th Annual Conference, TAMC 2017, Bern,* *Switzerland, April 20-22, 2017, Proceedings*, volume 10185 of *Lecture Notes in Computer Science*, pages 455–469, 2017.