WG1 Year-End Report
COST Action IC1405 Reversible Computation

Editors: Iain Phillips and Michael Kirkedal Thomsen

May 2019

List of contributors

Kamila Barylska
Tim Boykett
Gabriel Ciobanu
Robert Glück
Robin Kaarsgaard
Maciej Koutny
Ivan Lanese
Claudio Antares Mezzina
Łukasz Mikulski
Torben Ægidius Mogensen
Mohammad Reza Mousavi
Rajagopal Nagarajan
Luca Paolini
Iain Phillips
Marcin Piatkowski
Kyriaki Psara
Irek Ulidowski
German Vidal
Thomas Worsch

Contents

1 Introduction

This report covers research carried out with the aid of COST Action IC1405 on Reversible Computation 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 'Algebra of Reversible Circuits', '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 Foundations in General

> [GY18] Robert Glück and Tetsuo Yokoyama. Special issue on reversible computing: foundations and software. *New Generation Computing*, 36(3):143–306, 2018

Reversible computing is an emerging field of computer science that has received increasing attention during the past years. The articles in this special issue reflect the broad spectrum of research on reversible computing, including topics such as programming languages and semantics, methods and algorithms, computation models and theoretical foundations. Reversible computing has derived from fundamental questions on inverting programs and computations. Inverse problems arise frequently in mathematics, science and engineering, but the corresponding problems in computer science are yet to be fully understood. This special issue aims at contributing to this long-term endeavor by providing a forum in which answers to important questions are presented in detail.

Robert Glück

3 Finite-State Computing Models

There are no publications to report.

4 Reversible Cellular Automata

> [MU16] Daniel Morrison and Irek Ulidowski. Direction-reversible self-timed cellular automata for delay-insensitive circuits. *J. Cellular Automata*, 12(1-2):101–120, 2016

We introduce in this paper a new Self-Timed Cellular Automaton capable of simulating reversible delay-insensitive circuits. In addition to a number of reversibility and determinism properties, our STCA exhibits direction-reversibility, where reversing the direction of a signal and running a circuit forwards is equivalent

to running the circuit in reverse. We define also several extensions of the STCA which allow us to realise three larger classes of delay-insensitive circuits, including parallel circuits. We then show which of the reversibility, determinism and direction-reversibility properties hold for these classes of circuits.

Irek Ulidowski

> [KSW18a] Jarkko Kari, Ville Salo, and Thomas Worsch. Sequentializing cellular automata. In *Cellular Automata and Discrete Complex Systems - 24th IFIP WG 1.5 International Workshop, AUTOMATA 2018, Ghent, Belgium, June 20-22, 2018, Proceedings*, volume 10875 of *Lecture Notes in Computer Science*, pages 72–87. Springer, 2018

We study the problem of sequentializing a cellular automaton without introducing any intermediate states, and only performing reversible permutations on the tape. We give a decidable characterization of cellular automata which can be written as a single left-to-right sweep of a bijective rule from left to right over an infinite tape.

The authors gratefully acknowledge partial support for this work by two short term scientific missions of the EU COST Action IC1405.

Thomas Worsch

> [Kar18] Jarkko Kari. Reversible cellular automata: From fundamental classical results to recent developments. *New Generation Comput.*, 36(3):145–172, 2018

A cellular automaton is a dynamical system on an infinite array of cells defined by a local update rule that is applied simultaneously at all cells. By carefully choosing the update rule, the global dynamics can be made information preserving. In this case, the cellular automaton is called reversible. In this article, we explain fundamental classical results concerning reversible cellular automata and discuss some more recent developments on selected topics. Classical results reviewed include the Curtis–Hedlund–Lyndon theorem, the Garden-of-Eden theorem and the invariance of uniform Bernoulli distribution under reversible cellular automata. We then describe several techniques to construct reversible cellular automata and a method to determine whether a given one-dimensional automaton is reversible. We present undecidability issues concerning reversible cellular automata and discuss three types of universality: computational universality, intrinsic universality, and physical universality. We finish with short notes about time symmetry, expansiveness, and conservation laws.

Robert Glück

5 Algebra of Reversible Circuits

> [BKS16] Tim Boykett, Jarkko Kari, and Ville Salo. *Strongly Universal Reversible Gate Sets*, pages 239–254. Springer International Publishing, Cham, 2016

[BKS17] Tim Boykett, Jarkko Kari, and Ville Salo. Finite generating sets for reversible gate sets under general conservation laws. *Theoretical Computer Science*, 2017. In press

It is well-known that the Toffoli gate and the negation gate together yield a universal gate set, in the sense that every permutation of $\{0,1\}^n$ can be implemented as a composition of these gates. Since every bit operation that does not use all of the bits performs an even permutation, we need to use at least one auxiliary bit to perform every permutation, and it is known that one bit is indeed enough. Without auxiliary bits, all even permutations can be implemented. We generalize these results to non-binary logic: If A is a finite set of odd cardinality then a finite gate set can generate all permutations of A^n for all n, without any auxiliary symbols. If the cardinality of A is even then, by the same argument as above, only even permutations of A^n can be implemented for large n, and we show that indeed all even permutations can be obtained from a finite universal gate set. We also consider the conservative case, that is, those permutations of A^n that preserve the weight of the input word. The weight is the vector that records how many times each symbol occurs in the word. It turns out that no finite conservative gate set can, for all n, implement all conservative even permutations of A^n without auxiliary bits. But we provide a finite gate set that can implement all those conservative permutations that are even within each weight class of A^n.

[BKS17] extends and generalises the conference paper [BKS16]. These papers answer a conjecture and develop an alternative proof for one result in the following paper, that is in the refereeing process for a mathematics journal.

[Boy15] Tim Boykett. Closed systems of invertible maps. *CoRR*, abs/1512.06813, 2015

We generalise clones, which are sets of functions $f : A^n \to A$, to sets of maps $f : A^n \to A^m$. We formalise this and develop language that we can use to speak about such maps. In particular we look at bijective mappings, which model the logical gates of reversible computation. Reversible computation is important for physical (e.g. quantum computation) as well as engineering (e.g. heat dissipation) reasons. We generalise Toffoli's seminal work on reversible computation to multiple valued logics. In particular, we show that some restrictions Toffoli found for reversible computation on alphabets of order 2 do not apply for odd order alphabets. For A odd, we can create all invertible mappings from the Toffoli 1- and 2-gates, demonstrating that we can realise all reversible mappings from four generators. We discuss various forms of closure, corresponding to various systems of permitted manipulations. This leads, amongst other things, to discussions about ancilla bits in quantum computation.

Tim Boykett

6 Theory of Programming Languages

[Mog16] Torben Ægidius Mogensen. *RSSA: a reversible SSA form*, pages 203–217. Springer, 2016

The SSA form (Static Single Assignment form) is used in compilers as an intermediate language as an alternative to traditional three-address code because code in SSA form is easier to analyse and optimize using data-flow analysis such as common-subexpression elimination, value numbering, register allocation and so on.

We introduce RSSA, a reversible variant of the SSA form suitable as an intermediate language for reversible programming languages that are compiled to reversible machine language. The main issues in making SSA reversible are the unsuitability for SSA of the reversible updates and exchanges that are traditional in reversible languages and the need for ϕ-nodes on both joins and splits of control-flow. The first issue is handled by making selected uses of a variable destroy the variable and the latter by adding parameters to labels.

We show how programs in the reversible intermediate language RIL can be translated into RSSA and discuss copy propagation, constant propagation and register allocation in the context of RSSA.

Torben Ægidius Mogensen

[GY16] Robert Glück and Tetsuo Yokoyama. A linear-time self-interpreter of a reversible imperative language. *Computer Software*, 33(3):108–128, 2016

A linear-time reversible self-interpreter in an r-Turing complete reversible imperative language is presented. The proposed imperative language has reversible structured control flow operators and symbolic tree-structured data (S-expressions). The latter data structures are dynamically allocated and enable reversible simulation of programs of arbitrary size and space consumption. As self-interpreters are used to show a number of fundamental properties in classic computability and complexity theory, the present study of an efficient reversible self-interpreter is intended as a basis for future work on reversible computability and complexity theory as well as programming language theory for reversible computing. Although the proposed reversible interpreter consumes superlinear space, the restriction of the number of variables in the source language leads to linear-time reversible simulation.

[AG16] Holger Bock Axelsen and Robert Glück. On reversible Turing machines and their function universality. *Acta Informatica*, 53(5):509–543, 2016

We provide a treatment of the reversible Turing machines (RTMs) under a strict function semantics. Unlike many existing reversible computation models, we distinguish strictly between computing the function $\lambda x.f(x)$ and computing the function $\lambda x.(x, f(x))$, or other injective embeddings of f. We reinterpret and

adapt a number of important foundational reversible computing results under this semantics. Unifying the results in a single model shows that, as expected (and previously claimed), the RTMs are robust and can compute exactly all injective computable functions. Because injectivity entails that the RTMs are not strictly Turing-complete w.r.t. functions, we use an appropriate alternative universality definition, and show how to derive universal RTMs (URTMs) from existing irreversible universal machines. We then proceed to construct a URTM from the ground up. This resulting machine is the first URTM which does not depend on a reversible simulation of an existing universal machine. The new construction has the advantage that the interpretive overhead of the URTM is limited to a (program dependent) constant factor. Another novelty is that the URTM can function as an inverse interpreter at no asymptotic cost.

> [AGK16] Holger Bock Axelsen, Robert Glück, and Robin Kaarsgaard. A classical propositional logic for reasoning about reversible logic circuits. In Jouko Väänänen, Åsa Hirvonen, and Ruy de Queiroz, editors, *Logic, Language, Information, and Computation. Proceedings*, volume 9803 of *Lecture Notes in Computer Science*, pages 52–67. Springer-Verlag, 2016

We propose a syntactic representation of reversible logic circuits in their entirety, based on Feynman's control interpretation of Toffoli's reversible gate set. A pair of interacting proof calculi for reasoning about these circuits is presented, based on classical propositional logic and monoidal structure, and a natural order-theoretic structure is developed, demonstrated equivalent to Boolean algebras, and extended categorically to form a sound and complete semantics for this system. We show that all strong equivalences of reversible logic circuits are provable in the system, derive an equivalent equational theory, and describe its main applications in the verification of both reversible circuits and template-based reversible circuit rewriting systems.

> [GY17] Robert Glück and Tetsuo Yokoyama. A minimalist's reversible while language. *IEICE Transactions on Information and Systems*, E100-D, 2017

The paper presents a small reversible language R-CORE, a structured imperative programming language with symbolic tree-structured data (S-expressions). The language is reduced to the core of a reversible language, with a single command for reversibly updating the store, a single reversible control-flow operator, a limited number of variables, and data with a single atom and a single constructor. Despite its extreme simplicity, the language is reversibly universal, which means that it is as powerful as any reversible language can be, while it is linear-time self-interpretable, and it allows reversible programming with dynamic data structures. The four-line program inverter for R-CORE is among the shortest existing program inverters, which demonstrates the conciseness of the language. The translator to R-CORE, which is used to show the formal properties of the language, is clean and modular, and it may serve as a model

for related reversible translation problems. The goal is to provide a language that is sufficiently concise for theoretical investigations. Owing to its simplicity, the language may also be used for educational purposes.

> [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.

> [CGHM18] Martin Holm Cservenka, Robert Glück, Tue Haulund, and Torben Æ. Mogensen. Data structures and dynamic memory management in reversible languages. In Jarkko Kari and Irek Ulidowski, editors, *Reversible Computation. Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 269–285. Springer-Verlag, 2018

We present a method for reversible dynamic memory management based on a reversible version of the Buddy Memory system. This method supports decoupled allocation and deallocation of variable-sized records and can be applied to any reversible language with heap storage. We demonstrate how these new capabilities allow for the direct realization of commonplace data structures such as trees, heaps and queues which until now has not been practical in a reversible language. Finally, we provide a definition of our method in the high-level reversible language Janus as well as a description of its fragmentation and garbage-generation characteristics. The reversible memory management system has been fully implemented and tested in a compiler for a reversible object-oriented programming language targeting the reversible assembly language PISA.

> [Mog18] Torben Ægidius Mogensen. Reversible garbage collection for reversible functional languages. *New Generation Comput.*, 36(3):203–232, 2018

Reversible functional languages have been proposed that use patterns symmetrically for matching and building data: A pattern used on the left-hand side of a function rule takes apart a data structure, and a pattern used on the right-hand side of a rule builds a data structure. When calling a function in reverse,

the meaning of patterns are reversed, so patterns on the right-hand side take apart data and patterns on the left-hand side build data. If using a pattern to build data creates a node with exactly one reference, a node that is taken apart must by symmetry also have exactly one reference. This implies linearity: A node that is built or taken apart has exactly one incoming reference. A recursive function can take apart one data structure while building two or more new copies, allowing multiple uses of values, but the new copies will also have one reference each. Linearity makes garbage collection simple: Whenever a node is taken apart by pattern matching, it is freed. The cost is that multiple uses of a value require deep copies. The reason for the linearity restriction is the symmetry between constructing and deconstructing nodes: Since constructing a node creates exactly one reference to this node, the inverse can only be applied if the reference count is exactly one. We can overcome this limitation if constructing a node can return a node with multiple references. We achieve this through maximal sharing: If a newly constructed node is identical to an already existing node, we return a pointer to the existing node (increasing its reference count) instead of allocating a new node with reference count one. This allows multiple references to nodes while retaining the symmetry of pattern matching and construction, and it allows values to be used multiple times without making deep copies. To avoid searching the entire heap for an identical node, we use hash-consing to restrict the search to a small segment of the heap. We estimate how large this segment needs to be to give a low probability of allocation failure with acceptable utilisation and support this estimate with experiments. We sketch how a functional program can be translated to use the memory manager, and we test the memory manager both with artificial test programs and a hand-compiled functional program.

> [GY19] Robert Glück and Tetsuo Yokoyama. Constructing a binary tree from its traversals by reversible recursion and iteration. *Information Processing Letters*, 147:32–37, 2019

We cast two algorithms to generate the inorder and preorder of the labels of a binary tree in the context of reversible computing nearly three decades after they were first examined in the light of program inversion. The reversible traversals directly define the inverse algorithms that reconstruct the binary tree and have the same linear time and space requirements as the traversals. A reversible while-language is extended with reversible recursion.

Robert Glück

> [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.

> [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.

> [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.

Luca Paolini

7 Model-based Testing

> [HMTT17] Robert M. Hierons, Mohammad Reza Mousavi, Michael Kirkedal Thomsen, and Uraz Cengiz Türker. Hardness of deriving invertible

sequences from finite state machines. In *SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, volume 10139 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2017

We have performed research on test-case generation for finite state machines. Our focus has been on making the test-case generation and more efficient by using reversible subset of transitions in a finite state machine. In particular, we have focused on developing efficient algorithms for state verification using unique input-output sequences.

Mohammad Reza Mousavi

8 Term Rewriting

[NPV18] Naoki Nishida, Adrián Palacios, and Germán Vidal. Reversible computation in term rewriting. *J. Log. Algebr. Meth. Program.*, 94:128–149, 2018

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.

[NV19] Naoki Nishida and Germán Vidal. Characterizing compatible view updates in syntactic bidirectionalization. In Mathias Soeken and Michael Kirkedal Thomsen, editors, *Reversible Computation - 11th International Conference, RC 2019, Lausanne, Switzerland, June 24-25, 2019, Proceedings*, Lecture Notes in Computer Science. Springer, 2019

Given a function that takes a *source* data and returns a *view*, bidirectionalization aims at producing automatically a new function that takes a modified view and returns the corresponding, modified source. In this paper, we consider simple first-order functional programs specified by (conditional) term rewrite systems.

Then, we present a bidirectionalization technique based on the injectivization and inversion transformations from [NPV18]. We also prove a number of relevant properties which ensure that changes in both the source and the view are correctly propagated and that no undesirable side-effects are introduced. Furthermore, we introduce the use of narrowing—an extension of rewriting that re-places matching with unification—to precisely characterize *compatible* (also called *in-place*) view updates so that the resulting bidirectional transformations are well defined. Finally, we discuss some directions for dealing with view updates that are not compatible.

Germán Vidal

9 Categorical models and semantics

> [KAG17] Robin Kaarsgaard, Holger Bock Axelsen, and Robert Glück. Join inverse categories and reversible recursion. *Journal of Logical and Algebraic Methods in Programming*, 87:33–50, 2017

Recently, a number of reversible functional programming languages have been proposed. Common to several of these is the assumption of totality, a property that is not necessarily desirable, and certainly not required in order to guarantee reversibility. In a categorical setting, however, faithfully capturing partiality requires handling it as additional structure. Recently, Giles studied inverse categories as a model of partial reversible (functional) programming. In this paper, we show how additionally assuming the existence of countable joins on such inverse categories leads to a number of properties that are desirable when modeling reversible functional programming, notably morphism schemes for reversible recursion, a †-trace, and algebraic ω-compactness. This gives a categorical account of reversible recursion, and, for the latter, provides an answer to the problem posed by Giles regarding the formulation of recursive data types at the inverse category level.

Robin Kaarsgaard

> [GPY17] E. Graversen, I.C.C. Phillips, and N. Yoshida. Towards a categorical representation of reversible event structures. In *Proceedings of the International Workshop on Programming Language Approaches to Concurrency- and Communication-cEntric Software (PLACES 16)*, volume 246 of *EPTCS*, pages 49–60, 2017

We study categories for reversible computing, focussing on reversible forms of event structures. Event structures are a well-established model of true concurrency. There exist a number of forms of event structures, including prime event structures, asymmetric event structures, and general event structures. More recently, reversible forms of these types of event structures have been defined. We formulate corresponding categories and functors between them. We show that products and co-products exist in many cases. In most work on reversible

computing, including reversible process calculi, a cause-respecting condition is posited, meaning that the cause of an event may not be reversed before the event itself. Since reversible event structures are not assumed to be cause-respecting in general, we also define cause-respecting subcategories of these event structures. Our longer-term aim is to formulate event structure semantics for reversible process calculi.

Iain Phillips

> [GK18a] 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

> [GK18b] Robert Glück and Robin Kaarsgaard. A categorical foundation for structured reversible flowchart languages: Soundness and adequacy. *Logical Methods in Computer Science*, 14(3), 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. 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 decisions to give a reversible representation of predicates and assertions. This leads to a categorical semantics for structured

reversible flowcharts, which we show to be computationally sound and adequate, as well as equationally fully abstract with respect to the operational semantics under certain conditions.

Robert Glück

> [Kaa19] Robin Kaarsgaard. Inversion, iteration, and the art of dual wielding. In Mathias Soeken and Michael Kirkedal Thomsen, editors, *Reversible Computation - 11th International Conference, RC 2019, Lausanne, Switzerland, June 24-25, 2019, Proceedings*, Lecture Notes in Computer Science. Springer, 2019

The humble † ("dagger") is used to denote two different operations in category theory: Taking the adjoint of a morphism (in dagger categories) and finding the least fixed point of a functional (in categories enriched in domains). While these two operations are usually considered separately from one another, the emergence of reversible notions of computation shows the need to consider how the two ought to interact. In the present paper, we wield both of these daggers at once and consider dagger categories enriched in domains. We develop a notion of a monotone dagger structure as a dagger structure that is well behaved with respect to the enrichment, and show that such a structure leads to pleasant inversion properties of the fixed points that arise as a result. Notably, such a structure guarantees the existence of fixed point adjoints, which we show are intimately related to the conjugates arising from a canonical involutive monoidal structure in the enrichment. Finally, we relate the results to applications in the design and semantics of reversible programming languages.

Robin Kaarsgaard

> [GPY19] E. Graversen, I.C.C. Phillips, and N. Yoshida. Towards a categorical representation of reversible event structures. *Journal of Logical and Algebraic Methods in Programming*, 104:16–59, 2019

We study categories for reversible computing, focussing on reversible forms of event structures. Event structures are a well-established model of true concurrency. There exist a number of forms of event structures, including prime event structures, asymmetric event structures, and general event structures. More recently, reversible forms of these types of event structure have been defined. We formulate corresponding categories and functors between them. We show that products and coproducts exist in many cases.

We define stable reversible general event structures and stable configuration systems, and we obtain an isomorphism between the subcategory of the former in normal form and the finitely enabled subcategory of the latter.

In most work on reversible computing, including reversible process calculi, a causality condition is posited, meaning that the cause of an event may not be reversed before the event itself. Since reversible event structures are not assumed to be causal in general, we also define causal subcategories of these event structures.

Iain Phillips

10 Petri Nets

[BMP+16] Kamila Barylska, Lukasz Mikulski, Marcin Piatkowski, Maciej Koutny, and Evgeny Erofeev. Reversing transitions in bounded Petri nets. In *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming, Rostock, Germany, September 28-30, 2016*, volume 1698 of *CEUR Workshop Proceedings*, pages 74–85. CEUR-WS.org, 2016

[BKMP16] Kamila Barylska, Maciej Koutny, Lukasz Mikulski, and Marcin Piatkowski. Reversible computation vs. reversibility in Petri nets. In *Reversible Computation - 8th International Conference, RC 2016, Bologna, Italy, July 7-8, 2016, Proceedings*, volume 9720 of *Lecture Notes in Computer Science*, pages 105–118. Springer, 2016

Papers [BMP+16] and [BKMP16] contain preliminary results on the possibility of reversing the effect of the execution of Petri net transitions without retaining the previous stable states of the system or preserving the exact execution order of the transitions (but maintaining their partial order).

In [BMP+16], the undecidability of the problem of maintaining the set of reachable states of a p/t-net unchanged (related to the reachability problem) has been proven. Not only this cannot always be done, but no universal method can be proposed for checking if it is feasible. At the same time, it is possible to verify the invariability of executable computations (such a problem is related to the coverability problem in p/t-nets).

In [BKMP16], the case of bounded p/t-nets has been considered. It was shown that the problem of maintaining the set of reachable states unchanged is not only decidable, but - with the use of finite sets of inverse transitions - for each instance of this problem the answer is positive. The proof is constructive, and the proposed procedure based on the original p/t-net increases (at most twice) the number of places. Potentially, the number of the constructed reverses might be large when using the proposed naive algorithm (in fact, it can be as big as the number of occurrences of a reversed transition in the reachability graph of the original p/t-net). However, the procedure can be optimised, and one can always compute a minimal set of reverses.

The above results have been obtained thanks to two successful Short Term Scientific Missions of the COST action.

Kamila Barylska, Maciej Koutny, Łukasz Mikulski, Marcin Piatkowski

[PP17] Anna Philippou and Kyriaki Psara. Reversible computation in Petri nets. Technical report, University of Cyprus, 2017

We have conducted a research that studies reversible computation in the context of Petri Nets and in particular explores the modeling of the three main strategies for reversing computation. Our aim has been to address the challenges of capturing the notions of backtracking, causal reversibility and out-of-causal-order

reversibility within the Petri Net framework, thus proposing a novel, graphical methodology for studying reversible models where actions can be executed in either direction. Our approach is based on the introduction of memories for transitions as well as a special treatment of tokens, that requires them to be persistent and to retain individuality in order to allow the reversal of transitions in or out-of-causal order. The expressive power and visual nature offered by Petri Nets coupled with reversible computation has the potential of providing an attractive setting for analyzing systems. Indeed, good models that can be easily understood and simulated, even by scientists with expertise outside Computer Science, can prove very useful to understand complex systems. Furthermore, the applicability of our framework has been illustrated with an example of a biochemical system and an example of a transaction-processing system that naturally embed reversible behavior.

Kyriaki Psara

> [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.

Lukasz Mikulski

> [PP18] Anna Philippou and Kyriaki Psara. Reversible computation in Petri nets. In *Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 84–101. Springer, 2018

Reversible computation is an unconventional form of computing where any executed sequence of operations can be executed in reverse at any point during computation. In this paper we propose a reversible approach to Petri nets

by introducing machinery and associ- ated operational semantics to tackle the challenges of the three main forms of reversibility, namely, backtracking, causal reversing and out-of- causal-order reversing. Our proposal concerns a variation of Petri nets where tokens are persistent and are distinguished from each other by an identity. Our design decisions are influenced by applications in biochemistry but the methodology can be applied to a wide range of problems that feature reversibility. We demonstrate the applicability of our ap- proach with an example of a biochemical system and an example of a transaction-processing system both featuring reversible behaviour.

Kyriaki Psara

> [BGM+18] Kamila Barylska, Anna Gogolinska, Lukasz Mikulski, Anna Philippou, Marcin Piatkowski, and Kyriaki Psara. Reversing computations modelled by coloured petri nets. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2018 Satellite event of the conferences: 39th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2018 and 18th International Conference on Application of Concurrency to System Design ACSD 2018, Bratislava, Slovakia, June 25, 2018.*, volume 2115 of *CEUR Workshop Proceedings*, pages 91–111. CEUR-WS.org, 2018

This work is a continuation of the research line initiated in [PP18].

In the paper [BGM+18] a structural way of translating reversing Petri nets (RPNs), a formalism that embeds the three main forms of reversibility (backtracking, causal reversing and out-of-causal-order reversing), to Coloured Petri Nets (CPNs), an extension of traditional Petri Nets, where tokens carry data values, was proposed. The translation into the CPN model uses additional places and transitions in order to capture the machinery employed in the RPN framework and demonstrates that the abstract model of RPNs, and thus the principles of reversible computation, can be emulated in CPNs. The transformation is presented on several exmaples in CPN Tools software, but can be automated and utilized for the analysis of reversible systems using software.

The above results have been obtained thanks to two successful Short Term Scientific Missions of the COST Action.

> [dFEKM19] David de Frutos Escrig, Maciej Koutny, and Lukasz Mikulski. Reversing steps in Petri nets. In *Petri Nets 2019, Proceedings*, 2019. To appear

The work by David de Frutos Escrig, Maciej Koutny and Łukasz Mikulski entitled Reversing Steps in Petri Nets discuss the problem of reversing the effect of the execution of groups of actions (steps).

Using Petri nets as a system model, concepts related to this new scenario, generalising notions used in the single action case is introduced. A number of properties which arise in the context of reversing of steps of executed transitions

in place/transition nets is then presented. Both positive and negative results, showing that dealing with steps makes reversibility more involved than in the sequential case, are obtained. In particular, the crucial difference between reversing steps which are sets and those which are true multisets is discussed in details.

The above results have been obtained thanks to three successful Short Term Scientific Missions of the COST Action.

Łukasz Mikulski

> [ML19] Lukasz Mikulski and Ivan Lanese. Reversing unbounded Petri nets. In *Petri Nets 2019, Proceedings*, 2019. To appear

In Petri nets, computation is performed by executing transitions. An effect-reverse of a given transition b is a transition that, when executed, undoes the effect of b. A transition b is reversible if it is possible to add enough effect-reverses of b so to always being able to undo its effect, without changing the set of reachable markings.

This paper studies the *transition reversibility problem*: in a given Petri net, is a given transition b reversible? We show that, contrarily to what happens for the subclass of bounded Petri nets, the transition reversibility problem is in general *undecidable*. We show, however, that the same problem is *decidable in relevant subclasses* beyond bounded Petri nets, notably including all Petri nets which are cyclic, that is where the initial marking is reachable from any reachable marking. We finally show that some non-reversible Petri nets can be restructured, in particular by adding new places, so to make them reversible, while preserving their behaviour.

Ivan Lanese

11 Process Calculi

> [GLMT17] Elena Giachino, Ivan Lanese, Claudio Antares Mezzina, and Francesco Tiezzi. Causal-consistent rollback in a tuple-based language. *J. Log. Algebr. Meth. Program.*, 88:99–120, 2017

> [GLMT15] Elena Giachino, Ivan Lanese, Claudio Antares Mezzina, and Francesco Tiezzi. Causal-consistent reversibility in a tuple-based language. In *PDP*, pages 467–475. IEEE Computer Society, 2015

We have studied the definition of a rollback operator in the coordination language muKlaim [GLMT17], continuing the work undertaken in [GLMT15]. The main new result is that such an operator satisfies a simple intuitive specification, namely that it is the smallest causal-consistent set of backward moves undoing the target action.

Ivan Lanese

[BDLd15] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Ugo de'Liguoro. Retractable contracts. In *PLACES*, volume 203 of *EPTCS*, pages 61–72, 2015

[BLd17] Franco Barbanera, Ivan Lanese, and Ugo de'Liguoro. Retractable and speculative contracts. In *COORDINATION*, LNCS. Springer, 2017. to appear

[BdL16] Franco Barbanera and Ugo de' Liguoro. A game interpretation of retractable contracts. In Alberto Lluch-Lafuente and José Proença, editors, *COORDINATION*, volume 9686 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2016

[BDLd15] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Ugo de'Liguoro. Retractable contracts. In *PLACES*, volume 203 of *EPTCS*, pages 61–72, 2015

We have continued the study of retractable contracts started in [BDLd15]. Main results [BLd17] are that both compliance and the subcontract relation are decidable in polynomial time, and that the dual of a contract always exists and has a simple syntactic characterization. Furthermore, the same results apply to a novel model of contracts featuring a speculative choice: all the options of the choice are explored concurrently, and the computation succeeds if at least one of the options is successful. In [BdL16], instead, we have proposed a three-party game-theoretic interpretation of retractable session contracts, namely the retractable contracts of [BDLd15]. In such an interpretation a client is retractable-compliant with a server if and only if there exists a winning strategy for a particular player in a game-theoretic model of contracts. Such a player can be looked at as a mediator, driving the choices in the retractable points.

Ivan Lanese

[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

[MMPY18] Doriana Medic, Claudio Antares Mezzina, Iain Phillips, and Nobuko Yoshida. A parametric framework for reversible pi-calculi. In *Proceedings Combined 25th International Workshop on Expressiveness in Concurrency and 15th Workshop on Structural Operational Semantics and 15th Workshop on Structural Operational Semantics, EXPRESS/SOS*, volume 276 of *EPTCS*, pages 87–103, 2018

[Mez18] Claudio Antares Mezzina. On reversibility and broadcast. In *Reversible Computation - 10th International Conference, RC 2018*, volume 11106 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 2018

Causally consistent reversibility relates reversibility in a concurrent system with causality. In CCS there exist just one notion of causality, which is induced by the syntax of the terms. When moving to more expressive calculi, such as π-calculus thins are more complex. In π-calculus there exists different notions of causality, which differ in the treatment of parallel extrusions of the same name. In [MMPY18] a uniform framework for reversible π-calculi is presented. The framework is parametric with respect to a data structure that stores information about an extrusion of a name. Different data structures yield different approaches to the parallel extrusion problem. We map three well-known causal semantics into our framework. We show that the (parametric) reversibility induced by our framework is causally consistent and prove a causal correspondence between an appropriate instance of the framework and Boreale and Sangiorgi's causal semantics. Broadcast is a powerful primitive of communication used to model several distributed systems from local area networks, including wireless systems and lately multi-agent systems. In [Mez18] we study the interplay between reversibility and broadcast, in the setting of CCS endowed with a broadcast semantics. We first show how it is possible to reverse broadcast in CCS and then show that the obtained reversibility is causally consistent. We show the applicability of the proposed calculus by modelling the consensus algorithm.

Claudio Antares Mezzina

[Lan18] Ivan Lanese. From reversible semantics to reversible debugging. In *Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2018

This paper presents a line of research in reversible computing for concurrent systems. This line of research started in 2004 with the definition of the first reversible extensions for concurrent process calculi such as CCS, and is currently heading to the production of practical reversible debuggers for concurrent languages such as Erlang. Main questions that had to be answered during the research include the following. Which is the correct notion of reversibility for

concurrent systems? Which history information needs to be stored? How to control the basic reversibility mechanism? How to exploit reversibility for debugging? How to apply reversible debugging to real languages?

Ivan Lanese

> [UPY18] Irek Ulidowski, Iain Phillips, and Shoji Yuen. Reversing event structures. *New Generation Computing*, 36(3):281–306, Jul 2018

Reversible computation has attracted increasing interest in recent years. In this paper, we show how to model reversibility in concurrent computation as realised abstractly in terms of event structures. Two different forms of event structures are considered, namely event structures defined by causation and prevention relations and event structures given by an enabling relation with prevention. We then show how to reverse the two kinds of event structures, and discuss causal as well as out-of-causal order reversibility.

Irek Ulidowski

> [GPY18] E. Graversen, I.C.C. Phillips, and N. Yoshida. Event structure semantics of (controlled) reversible CCS. In *Proceedings of Tenth International Conference on Reversible Computation (RC 2018)*, volume 11106 of *Lecture Notes in Computer Science*, pages 102–122, 2018

CCSK is a reversible form of CCS which is causal, meaning that actions can be reversed if and only if each action caused by them has already been reversed; there is no control on whether or when a computation reverses. We propose an event structure semantics for CCSK. For this purpose we define a category of reversible bundle event structures, and use the causal subcategory to model CCSK. We then modify CCSK to control the reversibility with a rollback primitive, which reverses a specific action and all actions caused by it. To define the event structure semantics of rollback, we change our reversible bundle event structures by making the conflict relation asymmetric rather than symmetric, and we exploit their capacity for non-causal reversibility.

Iain Phillips

12 Membrane Computing

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

> [AC18b] 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 [AC18b] 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.

> [AC18a] Bogdan Aman and Gabriel Ciobanu. Bonding calculus. *Natural Computing*, 17(4):823–832, 2018

Process calculi, originally designed for modelling concurrent computations, have been widely applied to model biochemical and biological systems. Each term in the description of a biosystem has a counterpart in the biophysical system, and the behaviour of the whole system is obtained by following the operational semantics of the process calculi. Aman and Ciobanu introduced in [5] a process calculus able to describe and manipulate the bonding compounds by using only bond and unbond actions. The simulations by using the software platform Uppaal [6] can describe the dynamics of these bonding systems, and so it is possible to test the validity of some underlying assumptions and to verify various properties of the dynamics of the systems expressed in bonding calculus. This bonding calculus is also suitable to capture the out-of-order locally controlled reversibility defined in [9].

In [3], Aman and Ciobanu investigated the reversibility of biochemical reactions in membrane computing. Membrane computing [10] deals with multisets of symbols processed in the compartments of a membrane structure according to some multiset rewriting rules. The new features 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. Several theoretical results are proved, including so-called loop results and the connections between the evolutions of these systems and their reversible extensions.

In [4], Aman and Ciobanu investigated the reversibility in reaction systems. Reaction systems [8] represent a qualitative model; they deal with sets rather than multisets, assuming that a certain resource is present in the system in a sufficient amount such that several reactions needing such a resource are not in conflict. There are proved some results, including an operational correspondence between reaction systems and rewriting theory which allows a translation of the reversible reaction systems into some rewriting systems executable in the rewriting engine Maude [7].

In [1], Agrogoroaiei and Ciobanu present reversibility in membrane systems as a form of duality (under the influence of category theory). A full description of this kind of reversibility in membrane systems is given in [2].

References

[1] O. Agrigoroaiei, G. Ciobanu. Dual P Systems. *Lecture Notes in Computer Science* **5391**, 95–107 (2009).

[2] O. Agrigoroaiei, G. Ciobanu. Reversing Computation in Membrane Systems. *Journal of Logic and Algebraic Programming* **79**, 278-288 (2010).

[3] B. Aman, G. Ciobanu. Reversibility in Parallel Rewriting Systems. *Journal of Universal Computer Science* **23**(7), 692-703 (2017).

[4] B. Aman, G. Ciobanu. Controlled Reversibility in Reaction Systems. *Lecture Notes in Computer Science* **10725**, 40-53 (2018).

[5] B. Aman, G. Ciobanu. Bonding Calculus. *Natural Computing* **17**(4), 823-832 (2018).

[6] G. Behrmann, A. David, K.G. Larsen. A Tutorial on Uppaal . *Lecture Notes in Computer Science* **3185**, 200-236 (2004).

[7] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C.L. Talcott. *All About Maude - A High Performance Logical Framework. How to Specify, Program and Verify Systems in Rewriting Logic*. Springer (2007).

[8] A. Ehrenfeucht, G. Rozenberg. Reaction Systems. *Fundamenta Informaticae* **75**(1), 263-280 (2007).

[9] S. Kuhn, I. Ulidowski. Local Reversibility in a Calculus of Covalent Bonding. *Science of Computer Programming* **151**, 18-47 (2018).

[10] G. Păun. Computing with Membranes, *Journal of Computer and System Sciences* **61**, 108-143 (2000).

Gabriel Ciobanu

[Pin17] G. Michele Pinna. Reversing steps in membrane systems computations. In Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa, and Claudio Zandron, editors, *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 245–261. Springer, 2017

The issue of reversibility in computational paradigms has gained interest in recent years. Membrane systems are a computational device inspired by the living cell. Each membrane of a system is equipped with a multiset of objects and a set of rules which consumes the objects in the membrane possibly sending multisets in the neighboring membranes, being these organized in a tree like fashion. All the instance of the applicable rules have to be applied in order to reach the next state.

In this paper we investigate how to *reverse* steps in membrane systems computations. The problem is that computation steps in membrane systems do not preserve all the information that has to be used when reversing them, in particular for each produced object the information whether this has been produced by a rule residing in the membrane or by one in a neighboring one is lost.

We try here to formalize the relevant information needed, coding the rules applied and which object they have produced in a *memory* which is a suitable labeled partial order. We show that the proposed approach enjoy the so called *loop lemma*, which basically assures that the undoing obtained by reversely applying rules is correct.

G. Michele Pinna

13 Formal Verification of Quantum Systems

[BKN15] Jaap Boender, Florian Kammüller, and Rajagopal Nagarajan. Formalization of quantum protocols using Coq. In *Proceedings 12th International Workshop on Quantum Physics and Logic, QPL 2015, Oxford, UK, July 15-17, 2015.*, pages 71–83, 2015

Quantum Information Processing, which is an exciting area of research at the intersection of physics and computer science, has great potential for influencing the future development of information processing systems. The building of practical, general purpose Quantum Computers may be some years into the future. However, Quantum Communication and Quantum Cryptography are well developed. Commercial Quantum Key Distribution systems are easily available and several QKD networks have been built in various parts of the world. The security of the protocols used in these implementations rely on information-theoretic proofs, which may or may not reflect actual system behaviour. Moreover, testing of implementations cannot guarantee the absence of bugs and errors. This paper presents a novel framework for modelling and verifying quantum protocols and their implementations using the proof assistant Coq. We provide a Coq library

for quantum bits (qubits), quantum gates, and quantum measurement. As a step towards verifying practical quantum communication and security protocols such as Quantum Key Distribution, we support multiple qubits, communication and entanglement. We illustrate these concepts by modelling the Quantum Teleportation Protocol, which communicates the state of an unknown quantum bit using only a classical channel.

> [WN16] David Windridge and Rajagopal Nagarajan. Quantum bootstrap aggregation. In *Quantum Interaction - 10th International Conference, QI 2016, San Francisco, CA, USA, July 20–22, 2016, Revised Selected Papers*, pages 115–121, 2016

We set out a strategy for quantizing attribute bootstrap aggregation to enable variance-resilient quantum machine learning. To do so, we utilise the linear decomposability of decision boundary parameters in the Rebentrost et al. Support Vector Machine to guarantee that stochastic measurement of the output quantum state will give rise to an ensemble decision without destroying the superposition over projective feature subsets induced within the chosen SVM implementation. We achieve a linear performance advantage, $O(d)$, in addition to the existing $O(\log(n))$ advantages of quantization as applied to Support Vector Machines. The approach extends to any form of quantum learning giving rise to linear decision boundaries.

> [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.

> [AGN18] Ebrahim Ardeshir-Larijani, Simon J. Gay, and Rajagopal Nagarajan. Automated equivalence checking of concurrent quantum systems. *ACM Transactions on Computational Logic*, 19(4):28:1–28:32, 2018

The novel field of quantum computation and quantum information has gathered significant momentum in the last few years. It has the potential to radically impact the future of information technology and influence the development of modern society. The construction of practical, general purpose quantum computers has been challenging, but quantum cryptographic and communication

devices have been available in the commercial marketplace for several years. Quantum networks have been built in various cities around the world and a dedicated satellite has been launched by China to provide secure quantum communication. Such new technologies demand rigorous analysis and verification before they can be trusted in safety- and security-critical applications. Experience with classical hardware and software systems has shown the difficulty of achieving robust and reliable implementations.

We present CCSq, a concurrent language for describing quantum systems, and develop verification techniques for checking equivalence between CCSq processes. CCSq has well-defined operational and superoperator semantics for protocols that are functional, in the sense of computing a deterministic input-output relation for all interleavings arising from concurrency in the system. We have implemented QEC (Quantum Equivalence Checker), a tool that takes the specification and implementation of quantum protocols, described in CCSq, and automatically checks their equivalence. QEC is the first fully automatic equivalence checking tool for concurrent quantum systems. For efficiency purposes, we restrict ourselves to Clifford operators in the stabilizer formalism, but we are able to verify protocols over all input states. We have specified and verified a collection of interesting and practical quantum protocols, ranging from quantum communication and quantum cryptography to quantum error correction.

> [WMN18] David Windridge, Riccardo Mengoni, and Rajagopal Nagarajan. Quantum error-correcting output codes. *International Journal of Quantum Information*, 16(8):1840003, 2018

Quantum machine learning is the aspect of quantum computing concerned with the design of algorithms capable of generalized learning from labeled training data by effectively exploiting quantum effects. Error-correcting output codes (ECOC) are a standard setting in machine learning for efficiently rendering the collective outputs of a binary classifier, such as the support vector machine, as a multi-class decision procedure. Appropriate choice of error-correcting codes further enables incorrect individual classification decisions to be effectively corrected in the composite output. In this paper, we propose an appropriate quantization of the ECOC process, based on the quantum support vector machine. We will show that, in addition to the usual benefits of quantizing machine learning, this technique leads to an exponential reduction in the number of logic gates required for effective correction of classification error.

Rajagopal Nagarajan

Rajagopal Nagarajan

References

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

[AC18a] Bogdan Aman and Gabriel Ciobanu. Bonding calculus. *Natural Computing*, 17(4):823–832, 2018.

[AC18b] 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.

[AG16] Holger Bock Axelsen and Robert Glück. On reversible Turing machines and their function universality. *Acta Informatica*, 53(5):509–543, 2016.

[AGK16] Holger Bock Axelsen, Robert Glück, and Robin Kaarsgaard. A classical propositional logic for reasoning about reversible logic circuits. In Jouko Väänänen, Åsa Hirvonen, and Ruy de Queiroz, editors, *Logic, Language, Information, and Computation. Proceedings*, volume 9803 of *Lecture Notes in Computer Science*, pages 52–67. Springer-Verlag, 2016.

[AGN18] Ebrahim Ardeshir-Larijani, Simon J. Gay, and Rajagopal Nagarajan. Automated equivalence checking of concurrent quantum systems. *ACM Transactions on Computational Logic*, 19(4):28:1–28:32, 2018.

[BdL16] Franco Barbanera and Ugo de' Liguoro. A game interpretation of retractable contracts. In Alberto Lluch-Lafuente and José Proença, editors, *COORDINATION*, volume 9686 of *Lecture Notes in Computer Science*, pages 18–34. Springer, 2016.

[BDLd15] Franco Barbanera, Mariangiola Dezani-Ciancaglini, Ivan Lanese, and Ugo de'Liguoro. Retractable contracts. In *PLACES*, volume 203 of *EPTCS*, pages 61–72, 2015.

[BGM+18] Kamila Barylska, Anna Gogolinska, Lukasz Mikulski, Anna Philippou, Marcin Piatkowski, and Kyriaki Psara. Reversing computations modelled by coloured petri nets. In *Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data 2018 Satellite event of the conferences: 39th International Conference on Application and Theory of Petri Nets and Concurrency Petri Nets 2018 and 18th International Conference on Application of Concurrency to System Design ACSD 2018, Bratislava, Slovakia, June 25, 2018.*, volume 2115 of *CEUR Workshop Proceedings*, pages 91–111. CEUR-WS.org, 2018.

[BKMP16] Kamila Barylska, Maciej Koutny, Lukasz Mikulski, and Marcin Piatkowski. Reversible computation vs. reversibility in Petri nets. In *Reversible Computation - 8th International Conference, RC 2016,*

Bologna, Italy, July 7-8, 2016, Proceedings, volume 9720 of *Lecture Notes in Computer Science*, pages 105–118. Springer, 2016.

[BKN15] Jaap Boender, Florian Kammüller, and Rajagopal Nagarajan. Formalization of quantum protocols using Coq. In *Proceedings 12th International Workshop on Quantum Physics and Logic, QPL 2015, Oxford, UK, July 15-17, 2015.*, pages 71–83, 2015.

[BKS16] Tim Boykett, Jarkko Kari, and Ville Salo. *Strongly Universal Reversible Gate Sets*, pages 239–254. Springer International Publishing, Cham, 2016.

[BKS17] Tim Boykett, Jarkko Kari, and Ville Salo. Finite generating sets for reversible gate sets under general conservation laws. *Theoretical Computer Science*, 2017. In press.

[BLd17] Franco Barbanera, Ivan Lanese, and Ugo de'Liguoro. Retractable and speculative contracts. In *COORDINATION*, LNCS. Springer, 2017. to appear.

[BMP+16] Kamila Barylska, Lukasz Mikulski, Marcin Piatkowski, Maciej Koutny, and Evgeny Erofeev. Reversing transitions in bounded Petri nets. In *Proceedings of the 25th International Workshop on Concurrency, Specification and Programming, Rostock, Germany, September 28-30, 2016*, volume 1698 of *CEUR Workshop Proceedings*, pages 74–85. CEUR-WS.org, 2016.

[Boy15] Tim Boykett. Closed systems of invertible maps. *CoRR*, abs/1512.06813, 2015.

[CGHM18] Martin Holm Cservenka, Robert Glück, Tue Haulund, and Torben Æ. Mogensen. Data structures and dynamic memory management in reversible languages. In Jarkko Kari and Irek Ulidowski, editors, *Reversible Computation. Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 269–285. Springer-Verlag, 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.

[dFEKM19] David de Frutos Escrig, Maciej Koutny, and Lukasz Mikulski. Reversing steps in Petri nets. In *Petri Nets 2019, Proceedings*, 2019. To appear.

[GK18a] Robert Glück and Robin Kaarsgaard. A categorical foundation for structured reversible flowchart languages. *Electr. Notes Theor. Comput. Sci.*, 336:155–171, 2018.

[GK18b] Robert Glück and Robin Kaarsgaard. A categorical foundation for structured reversible flowchart languages: Soundness and adequacy. *Logical Methods in Computer Science*, 14(3), 2018.

[GLMT15] Elena Giachino, Ivan Lanese, Claudio Antares Mezzina, and Francesco Tiezzi. Causal-consistent reversibility in a tuple-based language. In *PDP*, pages 467–475. IEEE Computer Society, 2015.

[GLMT17] Elena Giachino, Ivan Lanese, Claudio Antares Mezzina, and Francesco Tiezzi. Causal-consistent rollback in a tuple-based language. *J. Log. Algebr. Meth. Program.*, 88:99–120, 2017.

[GPY17] E. Graversen, I.C.C. Phillips, and N. Yoshida. Towards a categorical representation of reversible event structures. In *Proceedings of the International Workshop on Programming Language Approaches to Concurrency- and Communication-cEntric Software (PLACES 16)*, volume 246 of *EPTCS*, pages 49–60, 2017.

[GPY18] E. Graversen, I.C.C. Phillips, and N. Yoshida. Event structure semantics of (controlled) reversible CCS. In *Proceedings of Tenth International Conference on Reversible Computation (RC 2018)*, volume 11106 of *Lecture Notes in Computer Science*, pages 102–122, 2018.

[GPY19] E. Graversen, I.C.C. Phillips, and N. Yoshida. Towards a categorical representation of reversible event structures. *Journal of Logical and Algebraic Methods in Programming*, 104:16–59, 2019.

[GY16] Robert Glück and Tetsuo Yokoyama. A linear-time self-interpreter of a reversible imperative language. *Computer Software*, 33(3):108–128, 2016.

[GY17] Robert Glück and Tetsuo Yokoyama. A minimalist's reversible while language. *IEICE Transactions on Information and Systems*, E100-D, 2017.

[GY18] Robert Glück and Tetsuo Yokoyama. Special issue on reversible computing: foundations and software. *New Generation Computing*, 36(3):143–306, 2018.

[GY19] Robert Glück and Tetsuo Yokoyama. Constructing a binary tree from its traversals by reversible recursion and iteration. *Information Processing Letters*, 147:32–37, 2019.

[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.

[HMTT17] Robert M. Hierons, Mohammad Reza Mousavi, Michael Kirkedal Thomsen, and Uraz Cengiz Türker. Hardness of deriving invertible sequences from finite state machines. In *SOFSEM 2017: Theory and Practice of Computer Science - 43rd International Conference on Current Trends in Theory and Practice of Computer Science, Limerick, Ireland, January 16-20, 2017, Proceedings*, volume 10139 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2017.

[Kaa19] Robin Kaarsgaard. Inversion, iteration, and the art of dual wielding. In Mathias Soeken and Michael Kirkedal Thomsen, editors, *Reversible Computation - 11th International Conference, RC 2019, Lausanne, Switzerland, June 24-25, 2019, Proceedings*, Lecture Notes in Computer Science. Springer, 2019.

[KAG17] Robin Kaarsgaard, Holger Bock Axelsen, and Robert Glück. Join inverse categories and reversible recursion. *Journal of Logical and Algebraic Methods in Programming*, 87:33–50, 2017.

[Kar18] Jarkko Kari. Reversible cellular automata: From fundamental classical results to recent developments. *New Generation Comput.*, 36(3):145–172, 2018.

[KSW18a] Jarkko Kari, Ville Salo, and Thomas Worsch. Sequentializing cellular automata. In *Cellular Automata and Discrete Complex Systems - 24th IFIP WG 1.5 International Workshop, AUTOMATA 2018, Ghent, Belgium, June 20-22, 2018, Proceedings*, volume 10875 of *Lecture Notes in Computer Science*, pages 72–87. Springer, 2018.

[KSW18b] 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.

[Lan18] Ivan Lanese. From reversible semantics to reversible debugging. In *Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 34–46. Springer, 2018.

[Mez18] Claudio Antares Mezzina. On reversibility and broadcast. In *Reversible Computation - 10th International Conference, RC 2018*, volume 11106 of *Lecture Notes in Computer Science*, pages 67–83. Springer, 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.

[ML19] Lukasz Mikulski and Ivan Lanese. Reversing unbounded Petri nets. In *Petri Nets 2019, Proceedings*, 2019. To appear.

[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.

[MMPY18] Doriana Medic, Claudio Antares Mezzina, Iain Phillips, and Nobuko Yoshida. A parametric framework for reversible pi-calculi. In *Proceedings Combined 25th International Workshop on Expressiveness in Concurrency and 15th Workshop on Structural Operational Semantics and 15th Workshop on Structural Operational Semantics, EXPRESS/SOS*, volume 276 of *EPTCS*, pages 87–103, 2018.

[Mog16] Torben Ægidius Mogensen. *RSSA: a reversible SSA form*, pages 203–217. Springer, 2016.

[Mog18] Torben Ægidius Mogensen. Reversible garbage collection for reversible functional languages. *New Generation Comput.*, 36(3):203–232, 2018.

[MU16] Daniel Morrison and Irek Ulidowski. Direction-reversible self-timed cellular automata for delay-insensitive circuits. *J. Cellular Automata*, 12(1-2):101–120, 2016.

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

[NPV18] Naoki Nishida, Adrián Palacios, and Germán Vidal. Reversible computation in term rewriting. *J. Log. Algebr. Meth. Program.*, 94:128–149, 2018.

[NV19] Naoki Nishida and Germán Vidal. Characterizing compatible view updates in syntactic bidirectionalization. In Mathias Soeken and Michael Kirkedal Thomsen, editors, *Reversible Computation - 11th International Conference, RC 2019, Lausanne, Switzerland, June 24-25, 2019, Proceedings*, Lecture Notes in Computer Science. Springer, 2019.

[Pin17] G. Michele Pinna. Reversing steps in membrane systems computations. In Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa, and Claudio Zandron, editors, *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 245–261. Springer, 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.

[PP17] Anna Philippou and Kyriaki Psara. Reversible computation in Petri nets. Technical report, University of Cyprus, 2017.

[PP18] Anna Philippou and Kyriaki Psara. Reversible computation in Petri nets. In *Reversible Computation - 10th International Conference, RC 2018, Leicester, UK, September 12-14, 2018, Proceedings*, volume 11106 of *Lecture Notes in Computer Science*, pages 84–101. Springer, 2018.

[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.

[UPY18] Irek Ulidowski, Iain Phillips, and Shoji Yuen. Reversing event structures. *New Generation Computing*, 36(3):281–306, Jul 2018.

[WMN18] David Windridge, Riccardo Mengoni, and Rajagopal Nagarajan. Quantum error-correcting output codes. *International Journal of Quantum Information*, 16(8):1840003, 2018.

[WN16] David Windridge and Rajagopal Nagarajan. Quantum bootstrap aggregation. In *Quantum Interaction - 10th International Conference, QI 2016, San Francisco, CA, USA, July 20–22, 2016, Revised Selected Papers*, pages 115–121, 2016.