COST Action IC1405
Reversible Computation
Extending Horizons of Computing

State of the Art Report
Working Group 1
Foundations

Editors:

Iain Phillips and Michael Kirkedal Thomsen

Department of Computing, Imperial College London,
United Kingdom
i.phillips@imperial.ac.uk

DIKU, Department of Computer Science,
University of Copenhagen, Denmark
kirkedal@acm.org

July 28, 2016

Chapter authors and contributors

- Holger Bock Axelsen, DIKU, Department of Computer Science, University of Copenhagen, Denmark
- Simon J. Gay, Department of Computing Science, University of Glasgow, United Kingdom
- Robin Kaarsgaard, DIKU, Department of Computer Science, University of Copenhagen, Denmark
- Jarkko Kari, Department of Mathematics, University of Turku, Finland
- Maciej Koutny, School of Computer Science, University of Newcastle, United Kingdom
- Martin Kutrib, Institut für Informatik, Universität Giessen, Germany
- Ivan Lanese, Dipartimento di Informatica, University of Bologna, Italy
- Łukasz Mikulski, School of Computer Science, University of Newcastle, United Kingdom
- Mohammad Reza Mousavi, Centre for Research on Embedded Systems (CERES), Halmstad University, Sweden
- Ville Salo, Department of Mathematics, University of Turku, Finland
- Michael Kirkedal Thomsen, DIKU, Department of Computer Science, University of Copenhagen, Denmark
- Francesco Tiezzi, Computer Science Division, School of Science and Technology, University of Camerino, Italy
- Irek Ulidowski, Department of Computer Science, University of Leicester, United Kingdom
- German Vidal, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain
- Thomas Worsch, Fakultät für Informatik, Karlsruhe Institute of Technology, Germany

Contents

Preface

This report presents the state of the art within theory and foundation of reversible computations. The work was initiated by Working Group 1 of the COST Action IC 1405 on reversible computations. The contents have been provided by the leading experts of the different research fields that are covered: see the Contents or each chapter for more details.

More on the project:

- `http://www.cost.eu/COST_Actions/ict/IC1405`
- `http://www.revcomp.eu/`

Chapter 1

Introduction

The search to understand the computation process and its limitations is as old as computers themselves. Here, we do not think of the algorithmic bounds (which is a very interesting topic by itself), but the limitations that are imposed by the physical world. All computers are situated in the physical world, and so the laws of physics, thus, also apply for computers including their circuits and memory. The question is what impact do the laws of physics have on the computation process?

During the 1950's (only shortly after the invention of the modern electronic digital computer [39]) the assumption arose that a logic operation requires an energy dissipation of $kT \ln 2$, where k is Boltzmann's constant and T is the temperature at which the operation is performed[1]. Von Neumann has later been credited for saying that this amount of energy is dissipated by two different sources: namely "per elementary decision of a two-way alternative and per elementary transmittal of one unit of information" [262][2]. From the first source, it *is* apparent that von Neumann meant something less than all computations and it sounds a lot like a conditional, which we today know to be a problem. Today, we also know that the second source does not necessitate energy dissipation (c.f. [162]).

Three foundational studies of reversible (or information lossless) computations can be dated back to the years around 1960. These studies were based on quite different computation models and motivations: Huffman studied information lossless finite state machines for their applications to data transmission [122], Landauer came to study reversible logic in his quest to determine the sources of energy dissipation in a computing system [161], and Lecerf studied reversible Turing machines for their theoretical properties [171]. Of these Landauer's work is most prominent, but the other two papers have laid the foundation that is today used in theoretical work.

Much has happened since this early work and this report will present the state of the art on the foundation and theory of reversible computations. First, in Chapter 2, we show results on the computational and descriptional complexity of finite-state computing models, after which, in Chapter 3, we present results

[1] This assumption followed from earlier work by Szilard [252] and Shannon [249] that argued that communicating one bit required this dissipation (c.f. [163]).

[2] John von Neumann died in 1957; four years before Landauer's seminal paper. The paper was finished by Burks and published in 1966, but the work still dates back to the 1950's.

on cellular automata: both synchronous and asynchronous automata. Following this, in Chapter 4 we give the foundational background on model-based testing that relates to reversibility and inversion. Chapter 5 describes research on reversible term rewriting and also touches on inversion. Next, Chapter 6, shows work from category theory that relates to reversibility exemplified by dagger and inverse categories. Chapter 7 shows recent research in reversibility in Petri nets, while Chapter 8 presents reversible process calculi and shows analysis techniques and applications. Finally, Chapter 9 discusses formal verification of quantum systems.

Foundational work on programming languages is also covered by this working group; however, it is not included in this report. The topic has a significant overlap to the work in *WG 2: Software and Systems* and it was therefore decided to include all work in their state of the art report. Work on the topic will still be a part of WG 1.

Chapter 2

Finite-State Computing Models: Computational and Descriptional Complexity

Holger Bock Axelsen[1,2] Martin Kutrib[3]

[1] Edlund A/S, Denmark

[2] DIKU, Department of Computer Science,
University of Copenhagen, Denmark
funkstar@diku.dk

[3] Institut für Informatik,
Universität Giessen, Germany
kutrib@informatik.uni-giessen.de

2.1 The Topic

Computers can be seen as information processing devices which are physical realizations of abstract computing models. From this viewpoint it is natural to study aspects of logical reversibility for finite-state machines with limited resources. One question of overriding importance is whether any computation (with respect to a given model) can be made reversible and for the costs of such a simulation, if possible at all.

For one of the standard universal computational models, the Turing machines, Bennett [29] showed the fundamental result that any computation of such a machine can be simulated reversibly. He proposed an efficient reversible simulation that uses $s \cdot t^{\log(3)}$ time and $s \cdot \log(t)$ space if the given machine takes t time and s space. However, for maximal t the space consumption is quadratic. In [170] a different method has been shown that uses almost no additional space but at the cost of exponential time. In [38] a general upper bound on the trade-off between time and space that suffices for the reversible simulation of

irreversible computations is proved, which reveals that it is possible to achieve sub-exponential time and sub-quadratic space simultaneously.

Since from a practical point of view universal machines are often too costly and even unnecessary for the task at hand, these results suggest to consider the power and costs for sub-universal reversible computing models. An important type of machines are finite-state devices with or without further resources as storage media (the access to which can be limited), limited parallelism, or restricted computational resources as time and space. Moreover, the costs for reversible simulations (and reversibility in general) have to be measured appropriately. For example, finite automata are models for control systems that are widely used in practice but they are not measured in terms of computational complexity resources such as time and space. On the other hand, they can be measured in terms of their sizes which builds a natural bridge to the costs for their implementations. Measuring the sizes of systems as opposed to computational resources means to consider their descriptional complexity. A survey on further aspects and concepts of reversibility in finite-state devices can be found, for example, in [150].

2.2 The Big Question: Two Sides of the Same Coin

Gaps: Which resources of finite-state computing models lead to a separation between reversibility and irreversibility?

In recent years, several models have been studied from this perspective. Bennett's results reveal that any resources strong enough to make a finite-state device universal render a separation impossible. This raises the question for devices with limited resources and, similarly, the interplay between such resources and the increase of computational complexity. For example, in [20] it is shown that the tape reduction of reversible Turing machines can be done at the same costs as for general Turing machines, so no extra costs are charged for reversibility with respect to this resource.

Restricting the space available for Turing machines does not lead to separations between reversibility and irreversibility, either [170]. On the other hand, we know of fine-grained time separations [21], but this is not particularly robust, and collapses under asymptotics. Limiting the space to an amount logarithmic in the length of the input yields the famous complexity class L. In particular, this class is characterized by finite-state devices that may read their input in parallel through some constant number of heads. It turned out that even this characterization carries over to the reversible world [19].

The latter model possesses a feature that unexpectedly closes the gap between reversibility and irreversibility. In order to characterize the complexity class L the machines have to be allowed to move their heads two-way. In [200] a reversible simulation of every irreversible two-way multi-head finite state machine is presented. On the other hand, restricting the head movements to one-way yields a gap between reversible and irreversible devices [152]. The same situation appears for finite automata. There are one-way finite automata that cannot be made reversible [236], but any two-way finite automaton can [143]. So,

going two-way is enough to close the gap between reversible and deterministic in many cases.

These results are complemented by the studies of reversible finite-state machines with additional storage, such as queues [153], pushdown stores [151], and input-driven (also known as visibly) pushdown stores [154].

Costs: What are the costs for reversible simulations of irreversible computations?

So far, only little is known about the costs. At the top of the hierarchy of finite-state devices the trade-offs between time and space for reversible simulations mentioned above are known. Moreover, the reduction from k tapes to 1 tape in general leads to a quadratic increase in time, and for k to 2 tapes, the time overhead is a logarithmic factor for deterministic as well as for reversible Turing machines.

On the opposite end of the hierarchy, there are the reversible finite-state machines without further resources. Several problems of their descriptional complexity have been solved in [121]. In particular, given a minimal deterministic finite automaton that can be made reversible, the size blow-up to an equivalent reversible finite automaton can be exponential. So, the costs for obtaining reversibility can be exorbitant. On the positive side, a minimal reversible automaton can effectively be constructed from the given deterministic one and it can algorithmically by decided whether a deterministic finite automaton can be made reversible.

2.3 The Future: Example Topics for Consideration in WG1

The discussion above suggests some natural and interesting topics for further research.

Descriptional complexity:

Almost nothing is known about the descriptional complexity of reversible finite-state devices apart from deterministic finite automata. Besides the questions discussed so far, so-called nonrecursive trade-offs are of particular interest. That is, the size trade-offs between different equivalent types of reversible devices are not bounded by any recursive function. In other words, the gain in economy of description can be arbitrary.

Computational complexity:

To what extent is the computational complexity increased when a given device is simulated reversibly, if it is possible at all? Are there fine-grained and robust time hierarchies? What about complexity measures beyond time and space? For example, in [155] reversible space-bounded Turing machines are considered that may rewrite any tape cell only in the first k visits, where k is a constant. Other interesting measures inlclude the number of head reversals, etc.

Degree of reversibility:

Assuming that a little bit of irreversibility is better than much irreversibility, the *degree* of reversibility of some devices can be considered. To this end, measures for the amount of (ir)reversibility have to be developed. A first approach is due to [12] and [157], where the degree is measured by the size of an input lookahead or input buffer that is necessary to perform reversible steps. What are reasonable measures? Are there infinite degree hierarchies? What relation, if any, can be made to general trade-offs for universal models?

Extended models:

We can also consider other ways to bridge the gap between reversible and irreversible machine. For instance, it is known that there are deterministic pushdown machines that cannot be made reversible [151]. We can boost the reversible pushdown (as well as finite automata) by slightly preprocessing the input by a reversible finite state machine in an injective and length-preserving way. In fact, the reversible machines with such a reversible preprocessing unit are strictly more powerful. So, a cheap extension of the classical models, in a way that does not 'morally' increase the irreversibility of the system, can have a dramatic impact. How about other such reversible extensions?

Reversible transducers:

Transducers are of great practical importance. Basically, these are classical devices that are equipped with some output unit. At the end of a successful computation the input is said to be translated to what has been emitted. So, it is natural to study reversible transducers as well. But how to define reversibility for transducers? What does reversibility of transducers mean?

Stronger notions of reversibility:

Physical reality reveals that often one can go back in time by applying the same transition function after a specific transformation of the phase-space. In [91] it is motivated that, for example, in Newtonian mechanics, relativity, or quantum mechanics one can go back in time by applying the same dynamics, provided that the sense of time direction is changed by a specific transformation of the phase-space. For Newtonian mechanics, the transformation leaves masses and positions unchanged but reverses the sign of the momenta. This aspect is called *time symmetry*. So, time-symmetric machines themselves cannot distinguish whether they run forward or backward in time. In this connection, computational models with discrete internal states, more precisely cellular automata, have been studied for the first time in [91]. Aspects of time-symmetry for reversible pushdown and finite-state machines have been considered in [156].

Are there other well-motivated strengthenings of reversibility? Are the devices that meet these notions strictly weaker?

Chapter Bibliography

[12] D. Angluin. Inference of reversible languages. *J. ACM*, 29(3):741–765, 1982

[19] H. B. Axelsen. Reversible multi-head finite automata characterize reversible logarithmic space. In A. H. Dediu and C. Martín-Vide, editors, *Language and Automata Theory and Applications (LATA 2012)*, volume 7183 of *LNCS*, pages 95–105. Springer, 2012

[20] H. B. Axelsen. Time complexity of tape reduction for reversible Turing machines. In A. D. Vos and R. Wille, editors, *Reversible Computation (RC 2011)*, volume 7165 of *LNCS*, pages 1–13. Springer, 2012

[21] H. B. Axelsen, S. Jakobi, M. Kutrib, and A. Malcher. A hierarchy of fast reversible Turing machines. In J. Krivine and J.-B. Stefani, editors, *Reversible Computation (RC 2015)*, volume 9138 of *LNCS*, pages 29–44. Springer, 2015

[29] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17:525–532, 1973

[38] H. Buhrman, J. Tromp, and P. M. B. Vitányi. Time and space bounds for reversible simulation. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *International Colloquium on Automata, Languages and Programming (ICALP 2001)*, volume 2076 of *LNCS*, pages 1017–1027. Springer, 2001

[91] A. Gajardo, J. Kari, and A. Moreira. On time-symmetry in cellular automata. *J. Comput. System Sci.*, 78:1115–1126, 2012

[121] M. Holzer, S. Jakobi, and M. Kutrib. Minimal reversible deterministic finite automata. In I. Potapov, editor, *Developments in Language Theory (DLT 2015)*, volume 9168 of *LNCS*, pages 276–287. Springer, 2015

[143] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *Foundations of Computer Science (FOCS 1997)*, pages 66–75. IEEE Computer Society, 1997

[150] M. Kutrib. Reversible and irreversible computations of deterministic finite-state devices. In G. F. Italiano, G. Pighizzini, and D. Sannella, editors, *Mathematical Foundations of Computer Science (MFCS 2015)*, volume 9234 of *LNCS*, pages 38–52. Springer, 2015

[151] M. Kutrib and A. Malcher. Reversible pushdown automata. *J. Comput. System Sci.*, 78:1814–1827, 2012

[152] M. Kutrib and A. Malcher. One-way reversible multi-head finite automata. In R. Glück and T. Yokoyama, editors, *Reversible Computation (RC 2012)*, volume 7581 of *LNCS*, pages 14–28. Springer, 2013

[153] M. Kutrib, A. Malcher, and M. Wendlandt. Reversible queue automata. In *Non-Classical Models of Automata and Applications (NCMA 2014)*, volume 304 of *books@ocg.at*, pages 163–178, Vienna, 2014. Austrian Computer Society

[154] M. Kutrib, A. Malcher, and M. Wendlandt. When input-driven pushdown automata meet reversibility. In R. Freund, M. Holzer, N. Moreira, and R. Reis, editors, *Non-Classical Models of Automata and Applications*

(NCMA 2015), volume 318 of *books@ocg.at*, pages 141–157, Vienna, 2015. Austrian Computer Society

[155] M. Kutrib and M. Wendlandt. Reversible limited automata. In J. Durand-Lose and B. Nagy, editors, *Machines, Computations, and Universality (MCU 2015)*, volume 9288 of *LNCS*, pages 113–128. Springer, 2015

[156] M. Kutrib and T. Worsch. Time-symmetric machines. In G. W. Dueck and D. M. Miller, editors, *Reversible Computation (RC 2013)*, volume 7948 of *LNCS*, pages 168–181. Springer, 2013

[157] M. Kutrib and T. Worsch. Degrees of reversibility for DFA and DPDA. In Y. Shigeru and M. Shin-ichi, editors, *Reversible Computation (RC 2014)*, volume 8507 of *LNCS*, pages 40–53. Springer, 2014

[170] K.-J. Lange, P. McKenzie, and A. Tapp. Reversible space equals deterministic space. *J. Comput. System Sci.*, 60:354–367, 2000

[200] K. Morita. Two-way reversible multi-head finite automata. *Fund. Inform.*, 110:241–254, 2011

[236] J.-E. Pin. On reversible automata. In *Latin 1992: Theoretical Informatics*, volume 583 of *LNCS*, pages 401–416. Springer, 1992

Chapter 3

Reversible Cellular Automata

Jarkko Kari[1] Thomas Worsch[2]

[1] Department of Mathematics,
University of Turku, Finland
jkari@utu.fi

[2] Fakultät für Informatik,
Karlsruhe Institute of Technology, Germany
worsch@kit.edu

With contributions by

- Ville Salo, Department of Mathematics, University of Turku, Finland
- Irek Ulidowski, Department of Computer Science, University of Leicester, United Kingdom

3.1 Synchronous CA

3.1.1 State of the art

A cellular automaton (CA) is a discrete dynamical system that consists of a regular grid of cells. Each cell has a state from a finite state set. The global state of the system at any given time is called its configuration. The cells change their states according to a local update rule that provides the new state based on the old states of the cell and its neighbors in the grid. All states use the same update rule, and in the classical synchronous situation the updating happens simultaneously at all cells. One update step of the cells induces a global function on configurations that describes a single step of the discrete time evolution of the system.

Basic facts:

- An injective global function is always surjective, hence bijective [197, 209].

- If Δ is bijective, then the inverse Δ^{-1} is a CA, too [110].

As far as synchronous CA are concerned the usual view of reversibility is: Each global configuration has at most one predecessor, i. e. the global transition function is bijective. There are other formulations which are obviously equivalent in the deterministic case, but are not equivalent when a global configuration can have more than one successor (e. g. for asynchronous CA).

Decidability

Already in 1972 it has been shown that for one-dimensional CA reversibility is decidable. The fastest decision algorithm known today needs quadratic time algorithm exists w.r.t. the rule table size [251]. Exact bounds on the neighborhood size of the inverse automaton are also known [46].

For two- and higher-dimensional CA the situation is radically different. It is undecidable if a given d-dimensional CA ($d \geq 2$) is reversible [130]. Therefore the inverse cellular automaton can have much larger neighborhood: the neighborhood cannot be bounded by a computable neighborhood.

Even for reversible one-dimensional cellular automata some problems are hard. It is for example undecidable [132] whether such a CA is periodic in the sense that from each initial configuration the CA returns to it after some time t.

Variants of the standard definition of CA

Since reversibility is undecidable for $d \geq 2$ dimensions, it is interesting to consider variants of the standard definition of CA. There are two related ideas to preserve the concept of only local interactions on one hand while on the other hand allow decidability of reversibility (in an almost trivial way).

One concept is to use block rules, which use the states of finite set of cells to specify the new states for all of them. The most prominent case is the so-called Margolus neighborhood [259]. It is known that every one- and two-dimensional reversible cellular automaton is a PCA when space is partitioned into sufficiently large blocks [131]. If the use of auxiliary bits is allowed, also higher-dimensional reversible CA can be simulated by reversible PCA [72]. But block PCA representations without auxiliary bits are not known in dimensions ≥ 3.

Partitioned CA [203] use the composition of local state permutations and partial shifts of information between neighbors). They often used in constructions providing reversible CA for which it has to be simple task to check that they are actually reversible (e. g. [202]).

Universality

It is known that there a computationally universal reversible cellular automata. There are several approaches for constructing them.

One is by simulating non-reversible ones by reversible ones. For example each d-dimensional non-reversible CA can be simulated by a $(d+1)$-dimensional reversible one [258]. Other constructions do not need to increase the number of dimensions but restrict themselves to configurations with finite support (e. g. [198] for one-dimensional CA) or change the notion of simulation [71]. As

a matter of fact the choice of the kind of simulation is important as can be seen by comparing [71] with [112].

Another approach to universality of reversible CA is by the simulation of reversible Turing machines which are known to be computationally universal [29]. For reversible one-dimensional PCA this has been shown in [203]. A more complicated construction [69] showed that even arbitrary Turing machines can be simulated in real time by reversible one-dimensional PCA.

There are also efforts to find reversible CA which are universal and satisfy additional properties, e. g. being number-conserving [202]. And once one is found, there are attempts to minimize constructions with respect to some descriptional complexity measures like the number of states or the number of neighbors.

3.1.2 Open problems

Examples of currently open problems include the following:

- Is every binary state reversible cellular automaton a composition of periodic ones? Or composition of involutions (Cellular Automata with period 2)?
- Is there a block PCA representation for each reversible cellular automaton even in $d \geq 3$ dimensions?

3.2 Asynchronous CA

3.2.1 State of the art

In asynchronous CA (ACA) a global configuration in general has more than one successor (and more than one predecessor). Instead of a one-step function one only has a one-step relation. ACA exhibit a restricted kind of nondeterminism. It is useless to speak about "at most one predecessor". The definition used and investigated in [263] is this: An ACA is reversible if the inverse of the one-step relation is again the one-step relation of an ACA. The restriction of this definition to synchronous CA coincides with the standard definition of reversibility.

Results [263] include the following which have to be compared with the synchronous case.

- Reversibility for d-dimensional ACA is decidable.
- There are computationally universal reversible ACA.

The above results need the technical detail that it is allowed that in one step *no* cell is active.

3.2.2 Open problems

- The role of block rules for ACA has not been investigated; the same is true for partitioned CA.

- If it is required that in each step at least one cell is active, then decidability for ACA is only known for the 1-dimensional case.
- If it is required that in each step at least one cell is active, then Turing completeness is not known.
- The results mentioned above hold for a certain definition (the "classical") definition of asynchronism. It is not known which can be carried over to other definitions.

3.3 Application of reversible cellular automata

Delay-insensitive (DI) networks are a category of asynchronous networks of modules (finite state machines) which make no assumption about delays within modules and lines connecting the modules, and have no global clock. It is argued in [187] that typical logical gates such as NAND and XOR are not Turing-complete when operated in a DI environment. Therefore implementations of DI modules and networks in alternative technologies, such as *cellular automata* [172], have been researched actively in recent years. DI networks were introduced by Keller [135] who characterised the conditions required for correct DI operation, and gave various universal sets.

Reversible DI modules and networks were studied originally by Adachi, Lee, Morita and Peper. They carried out research into finding efficient universal sets of reversible serial DI modules with memory, such as *Rotary Element* [199], and *Reading Toggle* and *Inverse Reading Toggle* [175]. The sets of all possible 2-state reversible serial modules with two, three and four pairs of input/output lines were enumerated in [204]. Implementations of reversible serial DI modules in *Self-Timed Cellular Automata* (STCAs) [221], are shown in several papers including [173–176]. How these various concepts relate to each other was discussed by Morita in [201]. Further investigations in [205,206] examine the effects of combining reversibility with parallelism in the context of DI modules. The new STCAs are proposed in [207] for the modelling of DI networks, including reversible parallel DI networks. They exhibit *direction-reversibility*, where reversing the direction of a signal and executing a network forwards is equivalent to executing the network in reverse.

3.4 Automorphisms of subshifts

3.4.1 Physical motivation for studying CA on subshifts

Suppose one wants to perform reversible computations on a piece P of some material X (e. g. a 1-dimensional string or an ingot of metal). Let's assume that P is completely uniform and its particles can be modified locally independently of each other. A natural simple formalization is to look at these pieces as standard d-dimensional CA.

Assume $S = \{0, 1\}$ is the set of states for each particle. There may be some restrictions for the material like two molecules in state 1 cannot occur next to each other. We then need to look at a subset $X \subset S^{\mathbb{Z}^d}$ of configurations. The

particular constraint where two 1s cannot occur next to each other is the so-called hard-core model in physics, and the golden mean or Fibonacci subshift in symbolic dynamics.

The question is then, *what reversible computations are possible for reversible CA, for different local constraints?* One point of interest is the fact that while computation can often be performed through simply encoding states of $S^{\mathbb{Z}}$ as states of X (by using multiple states to simulate one), reversibility very easily breaks in such a simulation.

3.4.2 A framework for studying cellular automata and other computational models 'as cellular automata'

The set $S^{\mathbb{Z}}$ of bi-infinite sequences over a finite alphabet S can be seen as a dynamical system where the time-evolution is given by the shift map. Cellular automata are the endomorphisms of this system, that is, they are the maps that preserve the topological and dynamical structure of $S^{\mathbb{Z}}$. Similarly, reversible cellular automata are the automorphisms of $S^{\mathbb{Z}}$. This means that we can see reversible CA as the 'symmetries' or 'rotations' of the mathematical object $S^{\mathbb{Z}}$. One can define similar notions on the multidimensional grids $S^{\mathbb{Z}^d}$.

Topologically and dynamically closed subsets of $S^{\mathbb{Z}}$ are called subshifts, and it is an interesting question what the automorphism groups of subshifts look like in general. In terms of cellular automata, the automorphism group of a subshift is the set of cellular automata that are well-defined and reversible on it. Automorphism groups of subshifts are a well-studied topic both from the mathematical and the computational point of view [35, 87, 110, 120, 244].

In [245], it is shown that there are strong connections between counter machines and endomorphisms of a class of subshifts called countable sofic shifts. In [27] it is shown that Turing machines can similarly be seen as automorphisms of subshifts. These observations give new and interesting connections between cellular automata and other computational models.

Further research in this direction for example leads to the insight that the so-called torsion problem is undecidable [132].

3.5 Further ideas for research

3.5.1 Fixing the neighborhood

Up to now research on reversibility for CA has assumed the size of the local neighborhood does not matter and may be chosen arbitrarily. It is known that the minimal neighborhood of the inverse of a CA may larger than that of the forward CA. This immediately implies that if one only allows the *same* neighborhood one gets a hierarchy of sets of problems solvable by reversibility CA, depending on the size of the neighborhood. Can one prove something?

3.5.2 Time-symmetry

A cellular automaton is time-symmetric if it is its own inverse on a suitably transformed phase-space [91]. More precisely, reversible CA G is time-symmetric if $G^{-1} = H \circ G \circ H$ for a CA H that is an involution, that is, H is its own inverse.

Then $G^{-n} = H \circ G^n \circ H$ for all n, so iterating G runs the system backwards in time on configurations transformed through symmetry H. The concept of time-symmetry was thoroughly studied in [91], but decidability of time-symmetry of a given one-dimensional CA remained an intriguing open problem.

3.5.3 Subshifts

The questions of whether all reversible CA can be decomposed into involutions or block PCA, asked above, are about finding generators for the automorphism group of $S^{\mathbb{Z}}$. There are many other open problems already about the automorphism group of $S^{\mathbb{Z}}$, including for example: Are the automorphism groups of $\{0,1\}^{\mathbb{Z}}$ and $\{0,1,2\}^{\mathbb{Z}}$ isomorphic? Is there a finitely generated subgroup of the automorphism group of $\{0,1\}^{\mathbb{Z}}$ where the torsion problem is undecidable?

The real interest of working with subshifts, however, is the study of reversible computation under local constraints, as in the physical motivation in Subsection 3.4.1.

- Are there sufficient dynamical conditions for embedding the automorphism group of $S^{\mathbb{Z}}$ in other subshifts? That is, under what restrictions can one simulate normal reversible CA under local constraints on the medium?
- Do automorphism groups of mixing SFTs embed in the automorphism group of $S^{\mathbb{Z}}$? That is, can we perfectly simulate cellular automata under any set of local constraints?
- In one dimension strong techniques for embedding the automorphism groups of $\{0,1\}^{\mathbb{Z}}$ and $\{0,1,2\}^{\mathbb{Z}}$ to each other [137] are known. In two dimensions there are geometric problems. Do the automorphism groups of $\{0,1\}^{\mathbb{Z}^2}$ and $\{0,1,2\}^{\mathbb{Z}^2}$ embed into each other?

3.5.4 Non-partitioned versus partitioned versus inverse partitioned CA

One step of a reversible PCA decomposes into two substeps: information exchange (or "partial shift") and local permutation. The inverse CA executes such operations in the reverse order, and hence is not automatically a PCA. Under what conditions is the inverse PCA again a PCA, and if it is a PCA, what about the size of the minimal neighborhood? One can also consider compositions and iterations of PCA and look for conditions for the result to be a PCA.

3.5.5 "Nonstandard" CA

Of course all questions addressed above can also be considered for other models which preserve the idea of only local communication but otherwise deviate from the standard definition of CA. These include reversibility for tree shaped CA, growing/shrinking CA, and parallel Turing machines [265].

Chapter Bibliography

[27] S. Barbieri, J. Kari, and V. Salo. *The Group of Reversible Turing Machines*, pages 49–62. Springer International Publishing, 2016

[29] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17:525–532, 1973

[35] M. Boyle, D. Lind, and D. Rudolph. The automorphism group of a shift of finite type. *Transactions of the American Mathematical Society*, 306(1):pp. 71–114, 1988

[46] E. Czeizler and J. Kari. A tight linear bound on the neighborhood of inverse cellular automata. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 410–420, 2005

[69] J.-C. Dubacq. How to simulate Turing machines by invertible one-dimensional cellular automata. *International Journal of Foundations of Computer Science*, 6(4):395–402, 1995

[71] J. O. Durand-Lose. Reversible space-time simulation of cellular automata. *Theoretical Computer Science*, 246:117–129, 2000

[72] J. O. Durand-Lose. Representing reversible cellular automata with reversible block cellular automata. In *Discrete Models: Combinatorics, Computation, and Geometry, DM-CCG 2001, Proceedings*, pages 145–154, 2001

[87] D. Fiebig and U.-R. Fiebig. The automorphism group of a coded system. *Transactions of the American Mathematical Society*, 348(8):3173–3191, 1996

[91] A. Gajardo, J. Kari, and A. Moreira. On time-symmetry in cellular automata. *J. Comput. System Sci.*, 78:1115–1126, 2012

[110] G. A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical Systems Theory*, 3:320–375, 1969

[112] P. Hertling. Embedding cellular automata into reversible ones. In C. S. Calude, J. Casti, and M. J. Dinneen, editors, *Unconventional Models of Computation, Proceedings*, pages 243–256. Springer, 1998

[120] M. Hochman. On the automorphism groups of multidimensional shifts of finite type. *Ergodic Theory Dynam. Systems*, 30(3):809–840, 2010

[130] J. Kari. Reversibility and surjectivity problems of cellular automata. *Journal of Computer and System Sciences*, 48(1):149–182, Feb. 1994

[131] J. Kari. Representation of reversible cellular automata with block permutations. *Mathematical Systems Theory*, 29(1):47–61, 1996

[132] J. Kari and N. Ollinger. Periodicity and immortality in reversible computing. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, pages 419–430, 2008

[135] R. Keller. Towards a theory of universal speed-independent modules. *IEEE Transactions on Computers*, 23(1):21–33, 1974

[137] K. H. Kim and F. W. Roush. On the automorphism groups of subshifts. *Pure Mathematics and Applications*, 1(4):203–230, 1990

[172] J. Lee, S. Adachi, F. Peper, and K. Morita. Embedding universal delay-insensitive circuits in asynchronous cellular spaces. *Fundamenta Informaticae*, 58(3-4):295–320, 2003

[173] J. Lee, S. Adachi, Y.-N. Xia, and Q.-S. Zhu. Emergence of universal global behavior from reversible local transitions in asynchronous systems. *Information Sciences*, 282:38 – 56, 2014

[174] J. Lee, X. Huang, and Q.-s. Zhu. Embedding simple reversed-twin elements into self-timed reversible cellular automata. *Journal of Convergence Information Technology*, 6(1), 2011

[175] J. Lee, F. Peper, S. Adachi, and K. Morita. An asynchronous cellular automaton implementing 2-state 2-input 2-output reversed-twin reversible elements. In *Proceedings of ACRI 2008*, volume 5191 of *LNCS*, pages 67–76. Springer, 2008

[176] J. Lee, F. Peper, S. Adachi, K. Morita, and S. Mashiko. Reversible computation in asynchronous cellular automata. In *Proceedings of Unconventional Models of Computation, UMC 2002*, pages 220–229, 2002

[187] A. J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In *Procs. of AUSCRIPT '90*, pages 263–278. MIT Press, 1990

[197] E. F. Moore. Machine models of self-reproduction. *Proceedings Symposium Applied Mathematics*, 14:17–33, 1963

[198] K. Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoretical Computer Science*, 148(1):157–163, 1995

[199] K. Morita. A simple universal logic element and cellular automata for reversible computing. In *Proceedings of MCU 2001*, volume 2055 of *LNCS*, pages 102–113. Springer, 2001

[201] K. Morita. Reversible computing systems, logic circuits, and cellular automata. In *Proceedings of Networking and Computing 2012*, pages 1–8, 2012

[202] K. Morita. Universality of one-dimensional reversible and number-conserving cellular automata. In *Proceedings 18th international workshop on Cellular Automata and Discrete Complex Systems and 3rd international symposium Journées Automates Cellulaires, AUTOMATA & JAC 2012*, pages 142–150, 2012

[203] K. Morita and M. Harao. Computation universality of one-dimensional reversible (injective) cellular automata. *IEICE Transactions (1976-1990)*, 72:758–762, 1989

[204] K. Morita, T. Ogiro, K. Tanaka, and H. Kato. Classification and universality of reversible logic elements with one-bit memory. In *Proceedings of MCU 2004*, volume 3354 of *LNCS*, pages 245–256. Springer, 2004

[205] D. Morrison and I. Ulidowski. Reversible delay-insensitive distributed memory modules. In *Proceedings of Reversible Computation 2013*, volume 7948 of *LNCS*, pages 11–24. Springer, 2013

[206] D. Morrison and I. Ulidowski. Arbitration and reversibility of parallel delay-insensitive modules. In *Proceedings of Reversible Computation 2014*, volume 8507 of *LNCS*, pages 67–81. Springer, 2014

[207] D. Morrison and I. Ulidowski. Direction-reversible self-timed cellular automata for delay-insensitive circuits. In *Proceedings of ACRI 2013*, volume 8751 of *LNCS*, pages 367–377. Springer, 2014

[209] J. Myhill. The converse to Moore's garden-of-eden theorem. *Proceedings of the American Mathematical Society*, 14:685–686, 1963

[221] F. Peper, T. Isokawa, N. Kouda, and N. Matsui. Self-timed cellular automata and their computational ability. *Future Generation Computer Systems*, 18(7):893–904, 2002

[244] V. Salo. Groups and monoids of cellular automata. In J. Kari, editor, *Cellular Automata and Discrete Complex Systems*, volume 9099 of *Lecture Notes in Computer Science*, pages 17–45. Springer Berlin Heidelberg, 2015

[245] V. Salo and I. Törmä. Computational aspects of cellular automata on countable sofic shifts. *Mathematical Foundations of Computer Science 2012*, pages 777–788, 2012

[251] K. Sutner. De Bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991

[258] T. Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15:213–231, 1977

[259] T. Toffoli and N. Margolus. *Cellular Automata Machines: A New Environment for Modeling*. MIT press, 1987

[263] S. Wacker and T. Worsch. On completeness and decidability of phase space invertible asynchronous cellular automata. *Fundamenta Informaticae*, 126(2-3):157–181, 2013

[265] J. Wiedermann. Weak parallel machines: A new class of physically feasible parallel machine models. In *Mathematical Foundations of Computer Science 1992, 17th International Symposium, MFCS 1992, Proceedings*, pages 95–111, 1992

Chapter 4

Model-based Testing

Mohammad Reza Mousavi[1] Michael Kirkedal Thomsen[2]

[1] Centre for Research on Embedded Systems (CERES),
Halmstad University, Sweden
m.r.mousavi@hh.se

[2] DIKU, Department of Computer Science,
University of Copenhagen, Denmark
kirkedal@acm.org

4.1 Model-Based Testing

Model-based testing ais at using test models for steering the test-case generation, execution, and analysis process. Several informal (or semi-formal) [191] and formal [36, 117] test models have been used for model-based testing. In this chapter, we mostly focus on formal test models and their role in model-based testing in the context of rerversibility. In this context, we distinguish three aspects regarding the role of reversibility: in test models, in test-case generation, and in test-case execution and analysis.

4.2 Reversible Test Models

In the context of reversible test models, we are only aware of reversible (or closely related) test models in the form of finite state machines. The concepts of reversibility are not in any way restricted to finite state machines, and other models can be define is similar notions of reversibility and inversion.

The first mention of reversible finite state machines was by Huffman [122], but the first use of FSMs in relation to testing was much later by Lukac et al. [185]. The two definitions differ slightly in that Huffman defines reversible FSMs to be forward and backward deterministic, while Lukac et al. defines them as balanced FSMs; for the completely defined the two definitions are identical.

Another model that lies between conventional FSMs and reversible FSMs is distinguished merging FSMs [106]. This model is (forward) deterministic while all ingoing transitions are unique when considering the input-output label.

Finally, there is also work on embedding conventional FSMs as reversible FSMs based on the Landauer (tracing) embedding [105].

4.3 Reversibility in Test-Case Generation

Test-case generation from a finite state machine model typically involves visiting each and every state of the test model and trying all its transitions. There are many different strategies for doing so [43, 243, 250] depending on the testing hypothesis, the fault model, and the test selection criteria. Different types of test sequences are used to bring the implementation to a state (e.g., using homing or synchronizing sequences) and check the current states of the implementation (e.g., using distinguishing or unique input/output sequences).

Both the computational complexity of finding such sequences and also the length of the generated sequences are important factors in test-case generation and any attempt in reducing either of them (or both) is potentially useful. Reversibility can come in handy in both cases: invertible sequences can be used to find overlap among different sequences and hence, shorten the total test sequence (by merging a few overlapping sequences) [115, 116]. Along the same lines, reversibility sequences may be used to efficiently generate some test sequences. For example, in [106] it is shown that for distinguishing merge FSMs, present distinguishing sequences (a type of state identification sequence) can be calculated in polynomial time, while in the general case the problem is PSPACE-complete. Another related idea pertains to unique input/output sequences and invertible sequences: once for a particular state s a unique input/output sequence is found, postfixes of an invertible sequence leading to s can be used as unique input/output for their source states in the invertible sequences [118].

Chapter Bibliography

[36] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner. *Model-Based Testing of Reactive Systems: Advanced Lectures*, volume 3472 of *Lecture Notes in Computer Science*. Springer, 2005

[43] T. S. Chow. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, May 1978

[105] M. Goudarzi, J. Chen, D. Vasudevan, E. Popovici, and M. Schellekens. Reversing deterministic finite state machines. In *Signals and Systems Conference (ISSC 2009), IET Irish*, pages 1–6, June 2009

[106] C. Güniçen, K. Inan, U. C. Türker, and H. Yenigün. The relation between preset distinguishing sequences and synchronizing sequences. *Formal Asp. Comput.*, 26(6):1153–1167, 2014

[115] R. M. Hierons. Extending test sequence overlap by invertibility. *Comput. J.*, 39(4):325–330, 1996

[116] R. M. Hierons. Testing from a finite-state machine: Extending invertibility to sequences. *Comput. J.*, 40(4):220–230, 1997

[117] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Lüttgen, A. J. H. Simons, S. Vilkomir, M. R. Woodward, and H. Zedan. Using Formal Specifications to Support Testing. *ACM Computing Surveys*, 41(2):9:1–9:76, 2009

[118] R. M. Hierons, M. R. Mousavi, M. Kirkedal Thomsen, and U. C. Türker. Hardness of deriving invertible sequences from finite state machines. Submitted

[122] D. A. Huffman. Canonical forms for information-lossless finite-state logical machines. *IRE Transactions on Information Theory*, 5(5):41–59, 1959

[185] M. Lukac, M. Kameyama, M. Perkowski, and P. Kerntopf. Analysis of reversible and quantum finite state machines using homing, synchronizing and distinguishing input sequences. In *Multiple-Valued Logic (ISMVL), 2013 IEEE 43rd International Symposium on*, pages 322–327, May 2013

[191] J. D. McGregor and D. A. Sykes. *A Practical Guide to Testing Object-oriented Software*. Object Technology Series. Addison Wesley, 2001

[243] K. Sabnani and A. Dahbura. A protocol test generation procedure. *Comput. Netw. ISDN Syst.*, 15(4):285–297, Sept. 1988

[250] A. Simão and A. Petrenko. Fault coverage-driven incremental test generation. *The Computer Journal*, 53:1508—1522, 2010

Chapter 5

Term Rewriting

German Vidal

Departamento de Sistemas Informáticos y Computación,
Universidad Politécnica de Valencia, Spain
gvidal@dsic.upv.es

With contributions by

- Irek Ulidowski, Department of Computer Science, University of Leicester, United Kingdom

5.1 Term Rewriting

Term rewriting [1, 23, 142] is a foundational theory of computing that underlies most rule-based programming languages. In this context, a *term rewriting system* (TRS) is a set of rewrite rules of the form $l \to r$ such that l is a nonvariable term and r is a term whose variables appear in l. Positions are used to address the nodes of a term viewed as a tree. A *position* p in a term t is represented by a finite sequence of natural numbers, where $t|_p$ denotes the *subterm* of t at position p and $t[s]_p$ the result of *replacing the subterm* $t|_p$ by the term s. *Substitutions* are mappings from variables to terms. For a TRS $\mathcal{R}$, we define the associated rewrite relation $\to_{\mathcal{R}}$ as the smallest binary relation satisfying the following: given terms s, t, we have $s \to_{\mathcal{R}} t$ iff there exist a position p in s, a rewrite rule $l \to r \in \mathcal{R}$, and a substitution σ such that $s|_p = l\sigma$ and $t = s[r\sigma]_p$.

The goal of term rewriting is reducing terms to so called *normal forms*, where a term t is called *irreducible* or in *normal form* w.r.t. a TRS $\mathcal{R}$ if there is no term s with $t \to_{\mathcal{R}} s$. Computing normal forms can be seen as the counterpart of computing *values* in functional programming.

5.2 Inverse Computation

Given an n-ary function f and an output v, an *inverse computation* aims at computing all the possible inputs $v_1, \ldots, v_n$ such that $f(v_1, \ldots, v_n) = v$ [241,

242]. Actually, this is often called *full* inverse computation, in contrast to *partial* computation where some inputs are also provided.

Traditionally, two approaches to inverse computation are distinguished [6]: *inverse interpreters* [6,63,189,246] that perform inverse computation taking the output v, the given inputs (if any), and the definition of f as input, and *inversion compilers* [11, 103, 104, 109, 134, 136, 190, 212, 215, 216, 241, 242] that performs *program inversion*. More precisely, inversion compilers take the definition of f as input and compute the definition of a (possibly partial) inverse function f^{-1}. *Semi-inversion* [194–196] is a more general notion than partial inversion that allows the original output to be partially given.

Typical applications of (full and partial) program inversion are the automatic development of dual programs: encryption and decryption (e.g., cryptographic encoder $\mathsf{enc}(x, k)$ and decoder $\mathsf{dec}(y, k)$ with a symmetric key k), data compression and decompression (e.g., zip/unzip), data/program translation and re-translation between different programming languages, and so on. Given one-half of dual programs, inversion provides the other program automatically and without bugs, thus guaranteeing high reliability.

The most popular target of program inversion is the class of injective functions (or functions that are injective w.r.t. the unknown arguments when *partial* inversion is considered). Deterministic definitions are expected as inverses of injective functions since the inverse relation for injective functions is one-to-one. However, this is not ensured by existing methods and in many cases overlapping and/or non-terminating functions are produced instead. Thus, the elimination of non-determinism in inverted functions has attracted a lot of interest [11, 103, 104, 134, 214].

In the context of term rewriting, a full-inversion method for so called *constructor* TRSs[1] has been proposed, and later extended to partial inversion in [212, 216]. The compiler (fully or partially) inverts a constructor TRS into a *conditional* term rewriting system (CTRS, for short) that completely defines inverses of functions defined in the original TRS. In a CTRS, the rules have the form $l \to r \Leftarrow s_1 \succ t_1, \ldots, s_n \succ t_n$, where each oriented equation $s_i \succ t_i$ is interpreted as reachability (i.e., $\to^*_{\mathcal{R}}$, the reflexive and transitive closure of $\to_{\mathcal{R}}$). The *conditional parts* of rewrite rules in the CTRS can be seen as `let`-structures for declaring variables that are locally used in the rewrite rules.

The method for eliminating non-determinism in [214], a post process for the compilers in [212, 216], consists in applying a restricted variant of *completion* to the systems obtained from the inverse conditional systems by *unraveling* [186, 218]. On the other hand, the method for eliminating non-determinism in [103,104,134] is based on applying LR parsing techniques to a grammar-based representation of functional programs. This is quite an interesting non-standard application of LR parsing and performs surprisingly well for the benchmarks of [133] that contain several kinds of schemes of function definitions (such as tail-recursion, non-tail-recursion, and the combination of both), despite the fact that only LR(0) parsing is considered.

In [217], a novel approach to program inversion which is specially tailored to deal with tail recursive functions defined by means of a rewrite system is introduced. This approach can be combined with the previous technique in [212]

[1] In a constructor TRS, the rules have the form $f(s_1, \ldots, s_n) \to r$ where f is a defined function and $s_1, \ldots, s_n$ are terms made up of data constructors and variables.

to produce a general inversion method.

5.3 Reversible Term Rewriting

One of the first approaches to reversibility in term rewriting is due to Abramsky [2], who considered reversibility in the context of *pattern matching automata*.[2] Abramsky's approach requires a condition called *biorthogonality* (which, in particular, implies injectivity), so that the considered automata are reversible. This work can be seen as a rather fundamental delineation of the boundary between reversible and irreversible computation in logical terms. However, biorthogonality is overly restrictive in the context of term rewriting, since almost no term rewrite system is biorthogonal.

More recently, Nishida *et al* [213] have introduced a *reversible* (but conservative) extension of term rewriting following a *Landauer's embedding*. In this approach, for every rewrite step $s \to_{\mathcal{R}} t$, one should store the applied rule β, the selected position p, and a substitution σ with the values of some variables (e.g., the variables that occur in the left-hand side of a rule but not in its right-hand side). Therefore, reversible rewrite steps have now the form $\langle s, \pi \rangle \rightharpoonup \langle t, \beta(p, \sigma) : \pi \rangle$, where $\rightharpoonup$ is a reversible (forward) rewrite relation and π is a *trace* that stores the sequence of terms of the form $\beta(p, \sigma)$. The dual, inverse relation $\leftharpoondown$ is also introduced, so that its union $\rightleftharpoons$ can be used to perform both forward and backward reductions.

Moreover, [213] also introduces an scheme to *compile* the reversible extension of rewriting into the system rules. Essentially, given a system $\mathcal{R}$, new systems $\mathcal{R}_f$ and $\mathcal{R}_b$ are produced, so that standard rewriting in $\mathcal{R}_f$, i.e., $\to_{\mathcal{R}_f}$, coincides with the forward reversible extension $\rightharpoonup_{\mathcal{R}}$ in the original system, and analogously $\to_{\mathcal{R}_b}$ is equivalent to $\leftharpoondown_{\mathcal{R}}$. Therefore, $\mathcal{R}_f$ can be seen as an *injectivization* of $\mathcal{R}$, and $\mathcal{R}_b$ can be seen as the *inversion* of $\mathcal{R}_f$. For this purpose, though, one should remove positions from traces since this information is *dynamic* (i.e., they depend on the term being reduced) and thus would require a complex (and inefficient) program instrumentation. For this purpose, a *flattening* transformation for CTRSs is introduced.

Finally, in order to illustrate the usefulness of reversible term rewriting, [3] presents applications in the fields of *bidirectional program transformation* (see, e.g., [88, 188] and references therein) and the reversibilization of *cellular automata* [4], so that a non-standard—i.e., a cellular automata with a potentially infinite number of states—reversible cellular automaton is obtained from an arbitrary one-dimensional cellular automaton, using the techniques from [213].

Another example of a term rewrite system with both forward and reverse rewrite relations is the *reaction systems for bonding* in [233]. It is used to model a simple catalytic reaction, polymer construction by a scaffolding protein and a long-running transaction with a compensation.

Chapter Bibliography

[1] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in The-*

[2] Although he did not consider rewriting explicitely, pattern matching automata can also be represented in terms of standard notions of term rewriting.

oretical Computer Science. Cambridge University Press, 2003

[2] Samson Abramsky. A structural approach to reversible computation. *Theor. Comput. Sci.*, 347(3):441–464, 2005

[3] Naoki Nishida, Adrián Palacios, and Germán Vidal. Reversible term rewriting in practice. In H. Cirstea and S. Escobar, editors, *Proc. of the Third International Workshop on Rewriting Techniques for Program Transformations and Evaluation (WPTE'16)*, 2016

[4] Kenichi Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theor. Comput. Sci.*, 148(1):157–163, 1995

[6] S. M. Abramov and R. Glück. The universal resolving algorithm and its correctness: inverse computation in a functional language. *Sci. Comput. Program.*, 43(2-3):193–229, 2002

[11] J. M. Almendros-Jiménez and G. Vidal. Automatic partial inversion of inductively sequential functions. In *Implementation and Application of Functional Languages, 18th International Symp osium, IFL 2006, Budapest, Hungary, September 4-6, 2006, Revised Selected Papers*, volume 4449 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2006

[23] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998

[63] N. Dershowitz and S. Mitra. Jeopardy. In *RTA*, pages 16–29, 1999

[88] N. Foster, K. Matsuda, and J. Voigtländer. Three complementary approaches to bidirectional programming. In J. Gibbons, editor, *Generic and Indexed Programming - International Spring School, SSGIP 2010, Oxford, UK, March 22-26, 2010, Revised Lectures*, volume 7470 of *Lecture Notes in Computer Science*, pages 1–46. Springer, 2012

[103] R. Glück and M. Kawabe. A program inverter for a functional language with equality and constructors. In *Proceedings of the first Asian Symposium on Programming Languages and Systems (APLAS 2003)*, pages 246–264, 2003

[104] R. Glück and M. Kawabe. A method for automatic program inversion based on LR(0) parsing. *Fundam. Inform.*, 66(4):367–395, 2005

[109] P. G. Harrison. Function inversion. In D. Bjørner, A. P. Ershov, and N. D. Jones, editors, *Proceedings of the International Workshop on Partial Evaluation and Mixed Computation*, pages 153–166. North-Holland, Amsterdam, 1988

[133] M. Kawabe and Y. Futamura. Case studies with an automatic program inversion system. In *Proceedings of the 21st Conference of Japan Society for Software Science and Technology*, number 6C-3, 5 pages, 2004

[134] M. Kawabe and R. Glück. The program inverter LRinv and its structure. In *Proceedings of the 7th International Symposium on Practical Aspects of Declarative Languages (PADL 2005)*, pages 219–234, 2005

[136] H. Khoshnevisan and K. M. Sephton. InvX: An automatic function inverter. In N. Dershowitz, editor, *Proceedings of the 3rd International Conference of Rewriting Techniques and Applications (RTA'89)*, volume 355 of *Lecture Notes in Computer Science*, pages 564–568. Springer, 1989

[142] J. W. Klop. Term Rewriting Systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume I, pages 1–112. Oxford University Press, 1992

[186] M. Marchiori. Unraveling and Ultraproperties. In M. Rodríguez-Artalejo and M. Hanus, editors, *Proc. of Sixth Int'l Conf. on Algebraic and Logic Programming, ALP'96*, pages 107–121. Springer LNCS 1139, 1996

[188] K. Matsuda, Z. Hu, K. Nakano, M. Hamana, and M. Takeichi. Bidirectionalization transformation based on automatic derivation of view complement functions. In R. Hinze and N. Ramsey, editors, *Proc. of the 12th ACM SIGPLAN International Conference on Functional Programming, ICFP 2007*, pages 47–58. ACM, 2007

[189] K. Matsuda, S.-C. Mu, Z. Hu, and M. Takeichi. A grammar-based approach to invertible programs. In *ESOP*, pages 448–467, 2010

[190] J. McCarthy. The inversion of functions defined by Turing machines. *Automata Studies*, pages 177–181, 1956

[194] T. Æ. Mogensen. Semi-inversion of guarded equations. In R. Glück and M. R. Lowry, editors, *Generative Programming and Component Engineering, 4th International Conference, GPCE 2005, Tallinn, Estonia, Proceedings*, volume 3676 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2005

[195] T. Æ. Mogensen. Report on an implementation of a semi-inverter. In I. Virbitskaite and A. Voronkov, editors, *Perspectives of Systems Informatics, 6th International Andrei Ershov Memorial Conference, PSI 2006. Revised Papers*, volume 4378 of *Lecture Notes in Computer Science*, pages 322–334. Springer, 2007

[196] T. Æ. Mogensen. Semi-inversion of functional parameters. In R. Glück and O. de Moor, editors, *Proceedings of the 2008 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation, PEPM 2008, San Francisco, California, USA*, pages 21–29. ACM, 2008

[212] N. Nishida. *Transformational Approach to Inverse Computation in Term Rewriting*. PhD thesis, Graduate School of Engineering, Nagoya University, Nagoya, Japan, Jan 2004

[213] N. Nishida, A. Palacios, and G. Vidal. Reversible term rewriting. In D. Kesner and B. Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal*, volume 52 of *LIPIcs*, pages 28:1–28:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016

[214] N. Nishida and M. Sakai. Completion after program inversion of injective functions. *Electr. Notes Theor. Comput. Sci.*, 237:39–56, 2009

[215] N. Nishida, M. Sakai, and T. Sakabe. Generation of inverse term rewriting systems for pure treeless functions. In Y. Toyama, editor, *Proceedings of the International Workshop on Rewriting in Proof and Computation (RPC'01), Sendai, Japan*, pages 188–198, Oct 2001

[216] N. Nishida, M. Sakai, and T. Sakabe. Partial inversion of constructor term rewriting systems. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA 2005)*, pages 264–278, 2005

[217] N. Nishida and G. Vidal. Program inversion for tail recursive functions. In *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA 2011)*, pages 283–298, 2011

[218] E. Ohlebusch. *Advanced topics in term rewriting*. Springer, 2002

[233] I. Phillips, I. Ulidowski, and S. Yuen. Modelling of bonding with processes and events. In *Proceedings of the 5th International Conference on Reversible Computation, RC 2013*, volume 7948 of *LNCS*, pages 141–154. Springer-Verlag, 2013

[241] S. A. Romanenko. A Compiler Generator Produced by a Self-Applicable Specializer Can Have a Surprisingly Natural and Understandable Structure. In D. Bjørner, A. P. Ershov, and N. D. Jones, editors, *Proceedings of the International Workshop on Partial Evaluation and Mixed Computation*, pages 445–463. North-Holland, Amsterdam, 1988

[242] S. A. Romanenko. Inversion and metacomputation. In *Partial Evaluation and Semantics-Based Program Manipulation*, pages 12–22. Sigplan Notices, 26(9), ACM, New York, September 1991

[246] J. P. Secher and M. H. Sørensen. From checking to inference via driving and dag grammars. In *PEPM*, pages 41–51. ACM, 2002. Also published in SIGPLAN Notices (vol. 37)

Chapter 6

Categorical models and categorical semantics

Robin Kaarsgaard

DIKU, Department of Computer Science,
University of Copenhagen, Denmark
robin@di.ku.dk

6.1 Category theory

Category theory is a framework for the description and development of mathematical structure, in which mathematical objects and their relationships in mathematical theories are abstracted into primal notions of *object* and *morphism*. Despite being a staple of the related field of quantum computer science for years (see, *e.g.*, [9, 125, 247]), category theory has seen comparatively little use in modelling reversible computation, where operational methods remain the standard.

6.2 Dagger categories

One approach to categorical models of reversible computation is given by †-categories, *i.e.*, categories with an abstract notion of inverse given by an involutive, identity-on-objects endofunctor $\dagger : \mathbf{C}^{\mathrm{op}} \to \mathbf{C}$. A useful specialization of these, in connection with reversible computation, is †-traced symmetric bimonoidal (or *rig*) categories, *i.e.*, †-categories equipped with two symmetric monoidal tensors (usually denoted $- \oplus -$ and $- \otimes -$) that interact appropriately (*e.g.*, $\otimes$ distributes over $\oplus$, the unit of $\oplus$ serves as annihilator for $\otimes$, and so on). Tail recursion is modelled by means of a trace operator (see [7, 129, 248]) that interacts with the †-functor in an appropriate way. These categories are strongly related to the †-compact closed categories [9, 247] that serve as the model of choice for the Oxford school of quantum computing.

The use of †-traced symmetric (bi)monoidal categories to model reversible computations goes back as least as far as to the works by Abramsky, Haghverdi

and Scott (see, *e.g.*, [8, 10]) on (reversible) combinatory algebras, though its applications in reversible programming was perhaps best highlighted by the development of the Π and Π^0 calculi [34,126]. In addition, the reversible functional programming language Theseus exhibits a correspondence with the Π^0 calculus, and thus with these categories.

Recent work by Heunen & Karvonen [113, 114], gives a †-categorical version of the correspondence between adjunctions and monads, as a correspondence between *Frobenius monads* and †-preserving adjoint functors. Given the history of monads in computer science, it seems interesting to explore how this theory may be applied, *e.g.*, in the study of reversible effects.

6.3 Inverse categories

Another approach to model reversible computation is inverse categories, a specialization of †-categories (alternatively, a specialization of restriction categories [44], or the categorical extension of inverse semigroups [119]) in which morphisms are required to be a partial isomorphism (intuitively, behave as partial injective functions).

In his somewhat recent thesis [101], B. Giles developed significant structural machinery intended for use in formalizing reversible programming languages, based on inverse category theory. This work, combined the comprehensive account of inverse categories with joins given in the thesis of Guo [107], has more recently been exploited by Axelsen & Kaarsgaard to give an account of reversible recursion in inverse categories with joins [22].

6.4 Groupoids

In recent work, Carette & Sabry [41] substitute the notion of a bimonoidal †-category with regards to Π [34] for that of a weak rig groupoid – *i.e.*, a category with a bimonoidal structure similar to bimonoidal dagger categories, but requiring that each morphism is a proper isomorphism, rather than simply having an abstract inverse. While this is arguably a stronger assumption, it allows them to show a pleasant correspondence, in the style of the Curry-Howard correspondence, between (certain) proof terms and (certain) reversible programs, by giving terms for certain second-order isomorphisms between first-order ones, constructed by way of the coherence theorem for rig categories.

An interesting open problem in this regard is whether this calculus can be exploited to give a finite, sound, and complete axiomatization of reversible logic circuits.

6.5 Other open problems

Though work has gone into developing both the appropriate theory, as well as reversible programming languages drawing heavy inspiration from categorical ideas, the gap between the two persists, in the sense that no work (to the knowledge of the author) has gone into developing (at least) sound and computationally adequate categorical models of reversible programming languages. This is a current topic of research at DIKU.

Chapter Bibliography

[7] S. Abramsky. Retracing some paths in process algebra. In U. Montanari and V. Sassone, editors, *CONCUR '96*, volume 1119 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 1996

[8] S. Abramsky. A structural approach to reversible computation. *Theoretical Computer Science*, 347(3):441–464, 2005

[9] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004*, pages 415–425. IEEE, 2004

[10] S. Abramsky, E. Haghverdi, and P. Scott. Geometry of Interaction and linear combinatory algebras. *Mathematical Structures in Computer Science*, 12(5):625–665, 2002

[22] H. B. Axelsen and R. Kaarsgaard. Join inverse categories as models of reversible recursion. In B. Jacobs and C. Löding, editors, *Foundations of Software Science and Computation Structures: 19th International Conference, FOSSACS 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 73–90. Springer, 2016

[34] W. J. Bowman, R. P. James, and A. Sabry. Dagger traced symmetric monoidal categories and reversible programming. In A. De Vos and R. Wille, editors, *Proceedings of RC 2011*, pages 51–56. Ghent University, 2011

[41] J. Carette and A. Sabry. Computing with semirings and weak rig groupoids. In P. Thiemann, editor, *Programming Languages and Systems: 25th European Symposium on Programming, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 123–148. Springer, 2016

[44] J. R. B. Cockett and S. Lack. Restriction categories I: Categories of partial maps. *Theoretical Computer Science*, 270(1–2):223–259, 2002

[101] B. G. Giles. *An investigation of some theoretical aspects of reversible computing.* PhD thesis, University of Calgary, 2014

[107] X. Guo. *Products, Joins, Meets, and Ranges in Restriction Categories.* PhD thesis, University of Calgary, 2012

[113] C. Heunen and M. Karvonen. Reversible monadic computing. *Electronic Notes in Theoretical Computer Science*, 319:217–237, 2015

[114] C. Heunen and M. Karvonen. Monads on dagger categories. arXiv preprint, arXiv:1602.04324, 2016

[119] P. M. Hines. *The Algebra of Self-Similarity and its Applications.* PhD thesis, University of Wales, Bangor, 1998

[125] B. Jacobs. New directions in categorical logic, for classical, probabilistic and quantum logic. *Logical Methods in Computer Science*, 11(3):1–76, 2015

[126] R. P. James and A. Sabry. Information effects. *ACM SIGPLAN Notices*, 47(1):73–84, 2012

[129] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996

[247] P. Selinger. Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical Computer Science*, 170:139–163, 2007

[248] P. Selinger. A survey of graphical languages for monoidal categories. In B. Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 289–355. Springer, 2011

Chapter 7

Petri Nets

Maciej Koutny Łukasz Mikulski

School of Computer Science,
University of Newcastle, United Kingdom
[maciej.koutny,lukasz.mikulski]@ncl.ac.uk

Petri nets [208,222,223,239] are a formal model of concurrent systems which supports both action-based and state-based modelling and reasoning. One of the important behavioural properties investigated in the context of Petri nets is reversibility, which in its basic formulation amounts to the possibility of returning to the initial marking (a global state) from any reachable marking (and thus ensuring that the net is cyclic). However, it is not required that any specific transitions or markings are used to bring the net back to the initial marking. Reversibility is a highly desirable feature in, e.g., hardware devices resetting themselves after a period of idleness, or self-stabilising systems recovering from failures. Application areas where it is crucial include embedded systems and flexible manufacturing systems. Reversibility is a fundamental behavioural property of Petri nets which is often considered in conjunction with liveness [208]. The latter guarantees that any transition (activity) can be executed, possibly after some delay, from every reachable marking. Live and reversible Petri nets exhibit therefore a kind of steady cyclic behaviour, and keep active all their functionalities.

Reversibility is often considered through more general *home states*, each home state being a marking which is reachable from any marking reachable from the initial one (hence a net is reversible whenever its initial marking is a home state). An even more general notion related to reversibility is that of home spaces, each home space being a set of markings such that from each reachable marking it is possible to reach at least one marking in this set (hence a marking is a home state if it forms a singleton home space).

The body of results on the decision problems related to reversibility and home states/spaces has been steadily growing over the past decades. These problems were usually considered within the domain of potentially infinite-state Place/Transition-net (PT-nets) and their subclasses, as most problems become

trivial for finite-state net models. Typically, these problems are of one of two kinds.

The first kind of problems concern establishing whether a given marking of set of markings satisfies a desirable property. For example, the fundamental home state problem, shown to be decidable in [57], is to establish whether a marking of a given PT-net is a home state. Its restricted version, consisting in deciding whether the initial marking of a PT-net is a home state was solved in [13]. Concerning the home spaces, [58] demonstrated that problem of establishing whether a linear set of markings is a home space of a given PT-net is decidable. Problems of the second kind ask whether there exists a marking or set of markings satisfies a desirable property. For example, the fundamental home state existence problem, shown to be decidable in [32], is to establish whether there exists a home state a given PT-net.

Although there are several positive decidability results related to reversibility, in general, the complexity of potential solutions appears to be high or difficult to establish. For example, the problem of the reversibility property is decidable but its complexity is still unknown [123], and [32] demonstrated that the problem of home state existence is at least as hard as the reachability problem [108]. These rather pessimistic results meant that the quest for effective algorithms, and indeed decidable problems, has for many years been carried out within special subclasses of PT-nets. Such subclasses, e.g. free-choice nets, are often defined by imposing restrictions on the structure of a Petri net, or assuming boundedness, with the resulting submodels of PT-nets being still relevant for a wide range of practical applications.

For example, it was shown in [33] that all live and bounded free-choice nets have home states, and the free-choice assumption cannot be changed to asymmetric choice. The home space problem is polynomial for live and bounded free-choice Petri nets [31, 64], and they also were shown to have home states [261]. Other, often progressively less restricted, net classes were considered in [31, 123, 238, 253, 254].

Reversibility has also been a topic of Petri net research on enforcing controllability in discrete event systems. Additional complementary places (called control places in [47] or monitor places in [240]) ensure reversibility to avoid deadlocks. The reversibility is closely linked with reachability and studied from the point of view of a specific class of Petri nets constructed with Elementary Control Tasks [86]. In [264] an elementary algorithm for detection of all home states for bounded nets is described. It is used to detect deadlocks (unique home states).

Chapter Bibliography

[13] T. Araki and T. Kasami. Decidable problems on the strong connectivity of Petri net reachability sets. *Theoretical Computer Science*, 4(1):99–119, 1977

[31] E. Best, J. Desel, and J. Esparza. Traps characterize home states in free choice systems. *Theoretical Computer Science*, 101(2):161–176, 1992

[32] E. Best and J. Esparza. Existence of home states in Petri nets is decidable. *Information Processing Letters*, 116(6):423–427, 2016

[33] E. Best and K. Voss. Free choice systems have home states. *Acta Informatica*, 21(1):89–100, 1984

[47] N. P. Danko Kezic and I. Petrovic. An algorithm for deadlock prevention based on iterative siphon control of Petri net. *Automatika*, 47(1-2):19–30, 2006

[57] D. de Frutos Escrig. Decidability of home states in place transition systems. Internal report. dpto. informatica y automatica, Univ. Complutense de Madrid, 1986

[58] D. de Frutos Escrig and C. Johnen. Decidability of home space property. Report LRI 503, Univ. de Paris-Sud, 1989

[64] J. Desel and J. Esparza. Reachability in cyclic extended free-choice systems. *Theoretical Computer Science*, 114(1):93–118, 1993

[86] L. Ferrarini. On the reachability and reversibility problems in a class of Petri nets. *IEEE Transactions on Systems, Man and Cybernetics*, 24(10):1474–1482, 1994

[108] M. Hack. *Decidability Questions for Petri Nets.* PhD thesis, MIT, 1975

[123] T. Hujsa, J.-M. Delosme, and A. M. Kordon. On the reversibility of live equal-conflict Petri nets. In R. R. Devillers and A. Valmari, editors, *Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings*, volume 9115 of *Lecture Notes in Computer Science*, pages 234–253. Springer, 2015

[208] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989

[222] J. L. Peterson. *Petri Net Theory and the Modeling of Systems.* Prentice Hall, 1 edition, 1981

[223] C. A. Petri. *Kommunikation mit Automaten.* PhD thesis, University of Bonn, 1962. (In German)

[238] L. Recalde, E. Teruel, and M. Silva. Modeling and analysis of sequential processes that cooperate through buffers. *IEEE Transactions on Robotics and Automation*, 14(2):267–277, 1998

[239] W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science.* Springer, 1982

[240] S. A. Reveliotis and J. Y. Choi. Designing reversibility-enforcing supervisors of polynomial complexity for bounded Petri nets through the theory of regions. In S. Donatelli and P. S. Thiagarajan, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2006, 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, Turku, Finland, June 26-30, 2006, Proceedings*, volume 4024 of *Lecture Notes in Computer Science*, pages 322–341. Springer, 2006

[253] E. Teruel, J. M. Colom, and M. Silva. Choice-free Petri nets: a model for deterministic concurrent systems with bulk services and arrivals. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 27(1):73–83, 1997

[254] E. Teruel and M. Silva. Liveness and home states in equal conflict systems. In M. A. Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 1993

[261] W. Vogler. Live and bounded free choice nets have home states. *Petri Net Newsletter*, 32:18–21, 1989

[264] P. Wang, Z. Ding, and H. Chai. An algorithm for generating home states of Petri nets. *Journal of Computational Information Systems*, 7(12):4225–4232, 2011

Chapter 8

Process Calculi

Ivan Lanese

Dipartimento di Informatica
University of Bologna, Italy
lanese@cs.unibo.it

With contributions by

- Francesco Tiezzi, Computer Science Division, School of Science and Technology, University of Camerino, Italy
- Irek Ulidowski, Department of Computer Science, University of Leicester, United Kingdom

8.1 Reversible Process Calculi

Process calculi are a class of algebraic models for concurrent and distributed systems.

Research on reversing process calculi can be perhaps tracked back to the *Chemical Abstract Machine* [30], a calculus inspired by chemical reactions whose operational semantics defines both forward and reverse reduction relations. The first attempts to reverse existing process calculi can be found in [48, 49], where a reversible extension of CCS [193] was presented. A main contribution of [49] was the definition of the notion of *causal-consistent reversibility*: any action can be undone, provided that its consequences, if any, are undone first. This definition is tailored to concurrent systems, where actions may overlap in time, hence saying "undo the last action" is not meaningful. Notably, this definition relates reversibility to *causality* instead of time, thus it can be applied even in those settings, such as some distributed systems, where no unique notion of time exists. A survey on causal-consistent reversibility can be found in [169].

Following [49], causal-consistent extensions of other and more expressive process calculi have been defined. They can be divided into two families, one dealing with calculi equipped with labeled transition system semantics (describing interactions between the process and the outside world), and one dealing

with reduction semantics (describing the evolution of processes in isolation). The first approach is more general, while the second is normally simpler and hence more easily applicable to expressive calculi. The first approach allowed the definition of extensions of CCS [48, 49], of calculi in a subset of the path format [225, 235], and of the π-calculus [45]. In the second line of research we find extensions of a fragment of CCS with biological relevance [40], of the higher-order π-calculus [166, 167], of Klaim [100], and of a π-calculus with sessions [255, 256]. A mapping from the instance on CCS of [225] to the CCS of [49] has been presented in [192].

This line of research concentrated on *uncontrolled* reversibility, defining how to reverse a process execution (in particular, which history and causal information is needed, and how to manage it), but not specifying when and whether to prefer backward execution over forward execution or vice versa. Uncontrolled reversibility allows one to understand how reversibility works, but not to exploit it into applications. Indeed, different application areas need different mechanisms to control reversibility. For instance, in biological systems the direction of the computation depends on physical conditions such as temperature and pressure, while in reliable systems reversibility is used to recover a consistent state when a bad event occurs. Triggered by these needs different mechanisms for *controlling* reversibility have been proposed. In particular, [50, 256] introduced irreversible actions to avoid going backward after a relevant result has been computed. [100, 164, 165, 168] proposed an explicit rollback operator undoing a past action inside calculi where normal computation is forward, and a mechanism of alternatives allowing one to avoid trying the same path again and again. [24] let an energy potential drive the direction of computation. [232] introduced a forward monitor controlling the direction of execution of a reversible monitored process. [148, 149] proposed a local mechanism where some past actions need to be undone in order to perform some new forward action. Interestingly, the two last approaches allow one to represent the so-called *out-of-causal order* reversibility, a form of reversibility that is common in biochemical reactions which does not preserve causal consistency. Out-of-causal order reversibility can be modelled in reversible event structures [229, 231].

The study of reversible process calculi triggered also research on more abstract models for describing concurrent systems. In particular, a series of papers [18,224,229,231,234,260] studied the impact of allowing events to be undone in addition to events taking place in different forms of event structures [266]. For example, prime event structures and asymmetric event structures are extended with reversible events in [229, 231] and reversible event structures defined with an enabling relation are introduced in [234]. Another work [51] showed that many of the results in [50] can be stated in a categorical framework.

8.2 Analysis Techniques and Applications

Given their formal definition, process calculi are suitable to formally verify properties of systems. Adapting the known techniques to reversible process calculi is not always easy, since one has to deal with history and causality information, while sometimes reversibility enables new techniques or results.

A main technique for process calculi is given by behavioral equivalences, allowing one to prove that two systems are equivalent from an external point

of view. This can be used to prove correctness of program transformations or optimizations. One of the earliest attempts attempts to study behavioral equivalences that were sensitive to reverse behavior was [28], where *back and forth bisimulation* was defined. Following this, and in response to an increased interest in reversible process calculi, behavioral equivalences based on bisimulations were studied in [52, 145, 224–226, 228, 235] and on barbed congruence in [18, 164, 165, 167]. A main point is that reverse computation (in a causal-consistent setting) increases the power of discrimination of these equivalences, since it allows one to distinguish actual concurrency (not creating causal dependencies) from nondeterminism (which establishes causal dependencies).

A similar phenomenon occurs in logics for reversible computation [227, 230], where reverse modalities, together with event identifiers, increase the expressive power of Hennessy-Milner style logics [111] on stable configuration structures to the point that the logics with reverse modalities characterise a range of bisimulation equivalences, including *history-preserving* bisimulation and *hereditary history-preserving* bisimulation [102], which are much more expressive than standard bisimulation, and many of the reverse bisimulations in [228].

Another line of work concerns session types and contracts (see the survey in [124]) for reversible systems. [256] shows how the session type discipline of π-calculus extends to its reversible variants. In [257], (binary and multiparty) session type systems are used to restrict the study of reversibility in π-calculus to single sessions. Instead, [25, 26] study the compliance of a client and a server when both of them have the ability to backtrack to past states.

Reversible calculi have born mainly with biological motivations, since many biological phenomena are naturally reversible, hence a reversible formalism seems suitable to model them. Indeed, efforts have been made to model biological systems [40, 48, 232, 234] as well as chemical reactions [148, 149] using reversible process calculi. We highlight [40] which illustrates a compilation from asynchronous CCS to DNA circuits.

In a different setting, reversible process calculi have also been used to build constructs for reliability, and in particular communicating transactions with compensations [59]. Transactions with compensations [37] are computations that either succeed, or their effects are compensated by a dedicated ad-hoc piece of code. In [59] the effect of the transaction is first undone, and then a compensation is executed. Behavioural equivalences for communicating transactions with compensation have also been studied [60, 144]. In [164] interacting transactions with compensations are mapped into a reversible calculus with alternatives.

Chapter Bibliography

[18] C. Aubert and I. Cristescu. Reversible barbed congruence on configuration structures. In *ICE*, volume 189 of *EPTCS*, pages 68–85, 2015

[24] G. Bacci, V. Danos, and O. Kammar. On the statistical thermodynamics of reversible communicating processes. In *CALCO*, volume 6859 of *LNCS*, pages 1–18. Springer, 2011

[25] F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro. Compliance for reversible client/server interactions. In *BEAT*, volume 162 of *EPTCS*,

pages 35–42, 2014

[26] F. Barbanera, M. Dezani-Ciancaglini, I. Lanese, and U. de'Liguoro. Retractable contracts. In *PLACES*, volume 203 of *EPTCS*, pages 61–72, 2015

[28] M. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical Report ICS PAS, Polish Academy of Sciences, 1991

[30] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1992

[37] R. Bruni, H. C. Melgratti, and U. Montanari. Theoretical foundations for compensations in flow composition languages. In *POPL*, pages 209–220. ACM, 2005

[40] L. Cardelli and C. Laneve. Reversible structures. In *CMSB*, pages 131–140. ACM, 2011

[45] I. Cristescu, J. Krivine, and D. Varacca. A compositional semantics for the reversible pi-calculus. In *LICS*, pages 388–397. IEEE Computer Society, 2013

[48] V. Danos and J. Krivine. Formal molecular biology done in CCS-R. In *BioConcur*, volume 180(3) of *ENTCS*, pages 31–49. Elsevier, 2003

[49] V. Danos and J. Krivine. Reversible communicating systems. In *CONCUR*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004

[50] V. Danos and J. Krivine. Transactions in RCCS. In *CONCUR*, volume 3653 of *LNCS*, pages 398–412. Springer, 2005

[51] V. Danos, J. Krivine, and P. Sobocinski. General reversibility. In *SOS*, volume 175(3) of *ENTCS*, pages 75–86. Elsevier, 2006

[52] V. Danos, J. Krivine, and F. Tarissan. Self-assembling trees. In *SOS*, volume 175(1) of *ENTCS*, pages 19–32. Elsevier, 2007

[59] E. de Vries, V. Koutavas, and M. Hennessy. Communicating transactions - (extended abstract). In *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 569–583. Springer, 2010

[60] E. de Vries, V. Koutavas, and M. Hennessy. Liveness of communicating transactions (extended abstract). In *APLAS*, volume 6461 of *LNCS*, pages 392–407. Springer, 2010

[100] E. Giachino, I. Lanese, C. A. Mezzina, and F. Tiezzi. Causal-consistent reversibility in a tuple-based language. In *PDP*, pages 467–475. IEEE Computer Society, 2015

[102] R. v. Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37:229–327, 2001

[111] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In *ICALP*, volume 85 of *LNCS*, pages 299–309. Springer, 1980

[124] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P.-M. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, and G. Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016

[144] V. Koutavas, C. Spaccasassi, and M. Hennessy. Bisimulations for communicating transactions - (extended abstract). In *FOSSACS*, volume 8412 of *LNCS*, pages 320–334. Springer, 2014

[145] J. Krivine. A verification technique for reversible process algebra. In *RC*, volume 7581 of *LNCS*, pages 204–217. Springer, 2012

[148] S. Kuhn and I. Ulidowski. Towards modelling of local reversibility. In *RC*, volume 9138 of *LNCS*, pages 279–284. Springer, 2015

[149] S. Kuhn and I. Ulidowski. A calculus for local reversibility. In *RC*, volume 9720 of *LNCS*, pages 20–35. Springer, 2016

[164] I. Lanese, M. Lienhardt, C. A. Mezzina, A. Schmitt, and J.-B. Stefani. Concurrent flexible reversibility. In *ESOP*, volume 7792 of *LNCS*, pages 370–390. Springer, 2013

[165] I. Lanese, C. A. Mezzina, A. Schmitt, and J.-B. Stefani. Controlling reversibility in higher-order pi. In *CONCUR*, volume 6901 of *LNCS*, pages 297–311, 2011

[166] I. Lanese, C. A. Mezzina, and J. Stefani. Reversibility in the higher-order π-calculus. *Theor. Comput. Sci.*, 625:25–84, 2016

[167] I. Lanese, C. A. Mezzina, and J.-B. Stefani. Reversing higher-order pi. In *CONCUR*, volume 6269 of *LNCS*, pages 478–493. Springer, 2010

[168] I. Lanese, C. A. Mezzina, and J.-B. Stefani. Controlled reversibility and compensations. In *RC*, volume 7581 of *LNCS*, pages 233–240. Springer, 2012

[169] I. Lanese, C. A. Mezzina, and F. Tiezzi. Causal-consistent reversibility. *Bulletin of the EATCS*, 114:121–139, 2014

[192] D. Medic and C. A. Mezzina. Static VS dynamic reversibility in CCS. In *RC*, volume 9720 of *LNCS*, pages 36–51. Springer, 2016

[193] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980

[224] I. Phillips and I. Ulidowski. Reversibility and models for concurrency. In *Essays on Algebraic Process Calculi*, volume 192(1) of *ENTCS*, pages 93–108, 2007

[225] I. Phillips and I. Ulidowski. Reversing algebraic process calculi. *J. Log. Algebr. Program.*, 73(1-2):70–96, 2007

[226] I. Phillips and I. Ulidowski. Reverse bisimulations on stable configuration structures. In *Proceedings of Structural Operational Semantics (SOS 2009)*, volume 18, pages 62–76. EPTCS, 2010

[227] I. Phillips and I. Ulidowski. A logic with reverse modalities for history-preserving bisimulations. In *EXPRESS*, volume 64 of *EPTCS*, pages 104–118, 2011

[228] I. Phillips and I. Ulidowski. A hierarchy of reverse bisimulations on stable configuration structures. *Mathematical Structures in Computer Science*, 22(2):333–372, 2012

[229] I. Phillips and I. Ulidowski. Reversibility and asymmetric conflict in event structures. In *CONCUR*, volume 8052 of *LNCS*, pages 303–318. Springer, 2013

[230] I. Phillips and I. Ulidowski. An event identifier logic. *Mathematical Structures in Computer Science*, 24:1–51, 2014

[231] I. Phillips and I. Ulidowski. Reversibility and asymmetric conflict in event structures. *Journal of Logical and Algebraic Methods in Programming*, 84(6):781–805, 2015

[232] I. Phillips, I. Ulidowski, and S. Yuen. A reversible process calculus and the modelling of the ERK signalling pathway. In *RC*, volume 7581 of *LNCS*, pages 218–232. Springer, 2012

[234] I. Phillips, I. Ulidowski, and S. Yuen. Modelling of bonding with processes and events. In *RC*, volume 7948 of *LNCS*, pages 141–154. Springer, 2013

[235] I. C. C. Phillips and I. Ulidowski. Reversing algebraic process calculi. In *FoSSaCS*, volume 3921 of *LNCS*, pages 246–260. Springer, 2006

[255] F. Tiezzi and N. Yoshida. Towards reversible sessions. In *PLACES*, volume 155 of *EPTCS*, pages 17–24, 2014

[256] F. Tiezzi and N. Yoshida. Reversible session-based pi-calculus. *J. Log. Algebr. Meth. Program.*, 84(5):684–707, 2015

[257] F. Tiezzi and N. Yoshida. Reversing single sessions. In *RC*, volume 9720 of *LNCS*, pages 52–69. Springer, 2016

[260] I. Ulidowski, I. Phillips, and S. Yuen. Concurrency and reversibility. In *RC*, volume 8507 of *LNCS*, pages 1–14. Springer, 2014

[266] G. Winskel. Event structures. In *Petri Nets: Central Models and Their Properties*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986

Chapter 9

Formal Verification of Quantum Systems

Simon J. Gay

Department of Computing Science,
University of Glasgow, United Kingdom
simon.gay@glasgow.ac.uk

We survey the main lines of work on formal verification of quantum information processing systems, especially communication protocols. The general theme of this work is to extend successful techniques from classical formal methods, by adding support for quantum computation and communication. The techniques can be classified under several headings: process calculus, model-checking, algebraic and graphical reasoning, other semantics-based approaches.

9.1 Formal Verification

The idea of formal verification is to prove the correctness of a system by means of well-founded analysis and reasoning. This is in contrast to testing, which is usually incomplete. A further characteristic of formal verification is that it should be systematic, and, ideally, supported by automated tools. In the domain of classical computing, many formal verification techniques and tools have been developed and applied. Inspired by this success, several groups of researchers have applied similar ideas to quantum computing and communication.

The starting point for formal verification is that the system being analysed needs to be defined or modelled in a formal language. In some cases this might be a practical programming language that is also used for implementation, but often there is a special-purpose modelling language which is simpler and more abstract. Process calculus is an example of such a modelling language, focussing on communication between concurrently executing components of a system. Successful classical process calculi include CCS, CSP and pi-calculus.

The next element is a formal specification of the desired behaviour of the system. In some approaches the specification is expressed in the same language

as the model of the system, and the goal of verification is to prove that the model and the specification are equivalent in the sense that they behave in the same way. This is known as *process-oriented specification*. An alternative approach is to express specifications in a separate logic, and then the goal of verification is to prove that the model of the system satisfies the logical properties representing the specification. This is known as *property-oriented specification*.

Finally, there needs to be a systematic way of checking that specifications are satisfied. This is based on a sound theoretical foundation: a theory of when two models are equivalent, or a theory of how a model satisfies a logical property. For small and simple systems it is possible to develop pen-and-paper proofs of correctness, according to a systematic approach, but for larger systems it is necessary to develop automated tools that can do the necessary analysis. These tools are known as model-checkers.

9.2 Quantum Process Calculus

Early work on quantum process calculus was done by Jorrand and Lalire, developing the QPAlg language and studying behavioural equivalence [127, 128, 158–160].

At around the same time, Gay and Nagarajan and co-authors investigated the analysis of quantum systems using the classical process calculus CCS and the classical probabilistic model-checker PRISM [95, 210, 211]. Subsequently they developed the quantum process calculus CQP, based on the pi-calculus [92, 93]. Later work by Davidson [53] developed a theory of behavioural equivalence, which has been used for manual analysis of several examples [55, 56]. Puthoor extended CQP to model linear optical quantum computing, taking account of the way in which qubits, treated as primitive in previous approaches, are implemented by dual-rail logic [89, 90, 98, 99, 237]. An overview of the work involving CQP appears in [94].

The most active line of work on quantum process calculus is by Ying and Feng and their co-authors [61,62,73–79,81–84,177,179,269–271] . Their language is qCCS, based on the classical process calculus CCS. The theory includes a thorough development of behavioural equivalences, including strong and weak bisimulation, as well as a notion of approximate bisimulation that supports quantitative reasoning about the degree of similarity of systems.

9.3 Model-Checking

Gay, Nagarajan and Papanikolaou developed QMC, a model-checking tool for property-oriented specification [96,97,219,220]. System models are expressed in a language similar to that of the classical probabilistic model-checker PRISM, and specifications are expressed in a quantum extension of temporal logic. Analysis is based on simulation of stabilizer quantum mechanics, and so systems are restricted to the use of Clifford group operators. This work has been linked to the work on process calculus by translating CQP into the language of the QMC model-checker [54].

Gay, Nagarajan and Ardeshir-Larijani [14–17] have developed a system for process-oriented model-checking, again based on stabilizer simulation and there-

fore also restricted to Clifford group operators. The modelling language is similar to qCCS. Models explicitly include communication between concurrent processes, and verification analyses all of the non-determinism resulting from alternative communication sequences.

Feng and co-authors have developed QPMC, a model-checker based on quantum Markov chains [80, 85, 178, 274].

9.4 Algebraic and Graphical Reasoning

Kubota *et al.* [146, 147, 267] have developed a system for automated reasoning about protocols expressed in qCCS, which is based on equational rewriting of algebraic expressions describing the quantum state. Although the rewriting is automated, suitable axioms must be put in by hand to capture the properties of quantum operators that are important for a particular protocol.

The Quantomatic tool [5, 65–68, 70, 138–141] is based on the diagrammatic presentation of categorical quantum mechanics. It uses partially-automated graph rewriting to verify process-oriented specifications: under human guidance, a graphical representation of a system or protocol is rewritten until it becomes a representation of the system's specification.

9.5 Other Semantics-Based Approaches

Ying and co-authors have developed theory for verification of quantum programs, with emphasis on algorithmic correctness in contrast to communication-based protocols [42, 180–183, 268, 272, 273, 275, 276]. This work includes the use of an automated theorem-prover to verify specifications expressed in quantum Hoare logic [184].

Chapter Bibliography

[5] The Quantomatic tool. `quantomatic.github.io`

[14] E. Ardeshir-Larijani. *Automated Equivalence Checking of Quantum Information Systems*. PhD thesis, University of Warwick, 2014

[15] E. Ardeshir-Larijani, S. J. Gay, and R. Nagarajan. Automated verification of quantum protocols by equivalence checking. *CoRR*, abs/1312.5951, 2013

[16] E. Ardeshir-Larijani, S. J. Gay, and R. Nagarajan. Equivalence checking of quantum protocols. In N. Piterman and S. A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7795 of *Lecture Notes in Computer Science*, pages 478–492. Springer, 2013

[17] E. Ardeshir-Larijani, S. J. Gay, and R. Nagarajan. Verification of concurrent quantum protocols by equivalence checking. In E. Ábrahám and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as*

Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings, volume 8413 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2014

[42] K. Chatterjee and J. Sgall, editors. *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, volume 8087 of *Lecture Notes in Computer Science*. Springer, 2013

[53] T. A. S. Davidson. *Formal Verification Techniques using Quantum Process Calculus*. PhD thesis, University of Warwick, 2012

[54] T. A. S. Davidson, S. J. Gay, H. Mlnarik, R. Nagarajan, and N. Papanikolaou. Model checking for communicating quantum processes. *IJUC*, 8(1):73–98, 2012

[55] T. A. S. Davidson, S. J. Gay, and R. Nagarajan. Formal analysis of quantum systems using process calculus. In A. Silva, S. Bliudze, R. Bruni, and M. Carbone, editors, *Proceedings Fourth Interaction and Concurrency Experience, ICE 2011, Reykjavik, Iceland, 9th June 2011.*, volume 59 of *EPTCS*, pages 104–110, 2011

[56] T. A. S. Davidson, S. J. Gay, R. Nagarajan, and I. V. Puthoor. Analysis of a quantum error correcting code using quantum process calculus. In B. Jacobs, P. Selinger, and B. Spitters, editors, *Proceedings 8th International Workshop on Quantum Physics and Logic, QPL 2011, Nijmegen, Netherlands, October 27-29, 2011.*, volume 95 of *EPTCS*, pages 67–80, 2011

[61] Y. Deng and Y. Feng. Open bisimulation for quantum processes. *CoRR*, abs/1201.0416, 2012

[62] Y. Deng and Y. Feng. Open bisimulation for quantum processes. In J. C. M. Baeten, T. Ball, and F. S. de Boer, editors, *Theoretical Computer Science - 7th IFIP TC 1/WG 2.2 International Conference, TCS 2012, Amsterdam, The Netherlands, September 26-28, 2012. Proceedings*, volume 7604 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2012

[65] L. Dixon and R. Duncan. Extending graphical representations for compact closed categories with applications to symbolic quantum computation. In S. Autexier, J. A. Campbell, J. Rubio, V. Sorge, M. Suzuki, and F. Wiedijk, editors, *Intelligent Computer Mathematics, 9th International Conference, AISC 2008, 15th Symposium, Calculemus 2008, 7th International Conference, MKM 2008, Birmingham, UK, July 28 - August 1, 2008. Proceedings*, volume 5144 of *Lecture Notes in Computer Science*, pages 77–92. Springer, 2008

[66] L. Dixon and R. Duncan. Graphical reasoning in compact closed categories for quantum computation. *Ann. Math. Artif. Intell.*, 56(1):23–42, 2009

[67] L. Dixon, R. Duncan, and A. Kissinger. Open graphs and computational reasoning. In S. B. Cooper, P. Panangaden, and E. Kashefi, editors, *Proceedings Sixth Workshop on Developments in Computational Models: Causality, Computation, and Physics, DCM 2010, Edinburgh, Scotland, 9-10th July 2010.*, volume 26 of *EPTCS*, pages 169–180, 2010

[68] L. Dixon and A. Kissinger. Open graphs and monoidal theories. *CoRR*, abs/1011.4114, 2010

[70] R. Duncan and M. Lucas. Verifying the steane code with quantomatic. In B. Coecke and M. J. Hoban, editors, *Proceedings of the 10th International Workshop on Quantum Physics and Logic, QPL 2013, Castelldefels (Barcelona), Spain, July 17-19, 2013.*, volume 171 of *EPTCS*, pages 33–49, 2013

[73] Y. Feng, Y. Deng, and M. Ying. Symbolic bisimulation for quantum processes. *CoRR*, abs/1202.3484, 2012

[74] Y. Feng, Y. Deng, and M. Ying. Symbolic bisimulation for quantum processes. *ACM Trans. Comput. Log.*, 15(2):14, 2014

[75] Y. Feng, R. Duan, Z. Ji, and M. Ying. Probabilistic bisimilarities between quantum processes. *CoRR*, abs/cs/0601014, 2006

[76] Y. Feng, R. Duan, Z. Ji, and M. Ying. Probabilistic bisimulations for quantum processes. *Inf. Comput.*, 205(11):1608–1639, 2007

[77] Y. Feng, R. Duan, and M. Ying. Bisimulation for quantum processes. *CoRR*, abs/1007.2584, 2010

[78] Y. Feng, R. Duan, and M. Ying. Bisimulation for quantum processes. In T. Ball and M. Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 523–534. ACM, 2011

[79] Y. Feng, R. Duan, and M. Ying. Bisimulation for quantum processes. *ACM Trans. Program. Lang. Syst.*, 34(4):17, 2012

[80] Y. Feng, E. M. Hahn, A. Turrini, and L. Zhang. QPMC: A model checker for quantum programs and protocols. In N. Bjørner and F. S. de Boer, editors, *FM 2015: Formal Methods - 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings*, volume 9109 of *Lecture Notes in Computer Science*, pages 265–272. Springer, 2015

[81] Y. Feng and M. Ying. Toward automatic verification of quantum cryptographic protocols. *CoRR*, abs/1507.05278, 2015

[82] Y. Feng and M. Ying. Toward automatic verification of quantum cryptographic protocols. In L. Aceto and D. de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 441–455. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015

[83] Y. Feng, N. Yu, and M. Ying. Model checking quantum markov chains. *CoRR*, abs/1205.2187, 2012

[84] Y. Feng, N. Yu, and M. Ying. Model checking quantum markov chains. *J. Comput. Syst. Sci.*, 79(7):1181–1198, 2013

[85] Y. Feng, N. Yu, and M. Ying. Reachability analysis of recursive quantum markov chains. In Chatterjee and Sgall [42], pages 385–396

[89] S. Franke-Arnold, S. J. Gay, and I. V. Puthoor. Quantum process calculus for linear optical quantum computing. In G. W. Dueck and D. M. Miller, editors, *Reversible Computation - 5th International Conference, RC 2013, Victoria, BC, Canada, July 4-5, 2013. Proceedings*, volume 7948 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2013

[90] S. Franke-Arnold, S. J. Gay, and I. V. Puthoor. Verification of linear optical quantum computing using quantum process calculus. In J. Borgström and S. Crafa, editors, *Proceedings Combined 21st International Workshop on Expressiveness in Concurrency and 11th Workshop on Structural Operational Semantics, EXPRESS 2014, and 11th Workshop on Structural Operational Semantics, SOS 2014, Rome, Italy, 1st September 2014.*, volume 160 of *EPTCS*, pages 111–129, 2014

[92] S. J. Gay and R. Nagarajan. Communicating quantum processes. In J. Palsberg and M. Abadi, editors, *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005, Long Beach, California, USA, January 12-14, 2005*, pages 145–157. ACM, 2005

[93] S. J. Gay and R. Nagarajan. Types and typechecking for communicating quantum processes. *Mathematical Structures in Computer Science*, 16(3):375–406, 2006

[94] S. J. Gay and R. Nagarajan. Techniques for formal modelling and analysis of quantum systems. In B. Coecke, L. Ong, and P. Panangaden, editors, *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky - Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, volume 7860 of *Lecture Notes in Computer Science*, pages 264–276. Springer, 2013

[95] S. J. Gay, R. Nagarajan, and N. Papanikolaou. Probabilistic model-checking of quantum protocols. *CoRR*, abs/quant-ph/0504007, 2005

[96] S. J. Gay, R. Nagarajan, and N. Papanikolaou. QMC: A model checker for quantum systems. In A. Gupta and S. Malik, editors, *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*, volume 5123 of *Lecture Notes in Computer Science*, pages 543–547. Springer, 2008

[97] S. J. Gay, R. Nagarajan, and N. Papanikolaou. Specification and verification of quantum protocols. In S. J. Gay and I. C. Mackie, editors, *Semantic Techniques in Quantum Computation*. Cambridge University Press, 2010

[98] S. J. Gay and I. V. Puthoor. Application of quantum process calculus to higher dimensional quantum protocols. In R. Duncan and P. Panangaden, editors, *Proceedings 9th Workshop on Quantum Physics and Logic, QPL*

2012, Brussels, Belgium, 10-12 October 2012., volume 158 of *EPTCS*, pages 15–28, 2014

[99] S. J. Gay and I. V. Puthoor. Equational reasoning about quantum protocols. In J. Krivine and J. Stefani, editors, *Reversible Computation - 7th International Conference, RC 2015, Grenoble, France, July 16-17, 2015, Proceedings*, volume 9138 of *Lecture Notes in Computer Science*, pages 155–170. Springer, 2015

[127] P. Jorrand and M. Lalire. From quantum physics to programming languages: A process algebraic approach. In J. Banâtre, P. Fradet, J. Giavitto, and O. Michel, editors, *Unconventional Programming Paradigms, International Workshop UPP 2004, Le Mont Saint Michel, France, September 15-17, 2004, Revised Selected and Invited Papers*, volume 3566 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004

[128] P. Jorrand and M. Lalire. Toward a quantum process algebra. In S. Vassiliadis, J. Gaudiot, and V. Piuri, editors, *Proceedings of the First Conference on Computing Frontiers, 2004, Ischia, Italy, April 14-16, 2004*, pages 111–119. ACM, 2004

[138] A. Kissinger. Exploring a quantum theory with graph rewriting and computer algebra. In J. Carette, L. Dixon, C. S. Coen, and S. M. Watt, editors, *Intelligent Computer Mathematics, 16th Symposium, Calculemus 2009, 8th International Conference, MKM 2009, Held as Part of CICM 2009, Grand Bend, Canada, July 6-12, 2009. Proceedings*, volume 5625 of *Lecture Notes in Computer Science*, pages 90–105. Springer, 2009

[139] A. Kissinger. Pictures of processes: Automated graph rewriting for monoidal categories and applications to quantum computing. *CoRR*, abs/1203.0202, 2012

[140] A. Kissinger and V. Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. *CoRR*, abs/1503.01034, 2015

[141] A. Kissinger and V. Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In A. P. Felty and A. Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 326–336. Springer, 2015

[146] T. Kubota, Y. Kakutani, G. Kato, Y. Kawano, and H. Sakurada. Automated verification of equivalence on quantum cryptographic protocols. In L. Kovács and T. Kutsia, editors, *5th International Symposium on Symbolic Computation in Software Science, SCSS 2013*, volume 15 of *EPiC Series*, pages 64–69. EasyChair, 2013

[147] T. Kubota, Y. Kakutani, G. Kato, Y. Kawano, and H. Sakurada. Semi-automated verification of security proofs of quantum cryptographic protocols. *J. Symb. Comput.*, 73:192–220, 2016

[158] M. Lalire. A probabilistic branching bisimulation for quantum processes. *CoRR*, abs/quant-ph/0508116, 2005

[159] M. Lalire. Relations among quantum processes: bisimilarity and congruence. *Mathematical Structures in Computer Science*, 16(3):407–428, 2006

[160] M. Lalire and P. Jorrand. A process algebraic approach to concurrent and distributed quantum computation: Operational semantics. *CoRR*, quant-ph/0407005, 2004

[177] L. Li and Y. Feng. Quantum markov chains: description of hybrid systems, decidability of equivalence, and model checking linear-time properties. *CoRR*, abs/1506.08982, 2015

[178] L. Li and Y. Feng. Quantum markov chains: Description of hybrid systems, decidability of equivalence, and model checking linear-time properties. *Inf. Comput.*, 244:229–244, 2015

[179] Y. Li and M. Ying. Debugging quantum processes using monitoring measurements. *CoRR*, abs/1403.4344, 2014

[180] Y. Li and M. Ying. (un)decidable problems about reachability of quantum systems. *CoRR*, abs/1401.6249, 2014

[181] Y. Li and M. Ying. (un)decidable problems about reachability of quantum systems. In P. Baldan and D. Gorla, editors, *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, volume 8704 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2014

[182] Y. Li, N. Yu, and M. Ying. Termination of nondeterministic quantum programs. *CoRR*, abs/1201.0891, 2012

[183] Y. Li, N. Yu, and M. Ying. Termination of nondeterministic quantum programs. *Acta Inf.*, 51(1):1–24, 2014

[184] T. Liu, Y. Li, S. Wang, M. Ying, and N. Zhan. A theorem prover for quantum hoare logic and its applications. *CoRR*, abs/1601.03835, 2016

[210] R. Nagarajan and S. J. Gay. Formal verification of quantum protocols. *arXiv*, quant-ph/0203086,, 2002

[211] R. Nagarajan, N. Papanikolaou, G. Bowen, and S. J. Gay. An automated analysis of the security of quantum key distribution. *CoRR*, abs/cs/0502048, 2005

[219] N. Papanikolaou. Techniques for design and validation of quantum protocols. Master's thesis, University of Warwick, 2004

[220] N. Papanikolaou. *Model Checking Quantum Protocols*. PhD thesis, University of Warwick, 2009

[237] I. V. Puthoor. *Theory and Applications of Quantum Process Calculus*. PhD thesis, University of Glasgow, 2015

[267] K. Yasuda, T. Kubota, and Y. Kakutani. Observational equivalence using schedulers for quantum processes. In B. Coecke, I. Hasuo, and P. Panangaden, editors, *Proceedings of the 11th workshop on Quantum Physics and Logic, QPL 2014, Kyoto, Japan, 4-6th June 2014.*, volume 172 of *EPTCS*, pages 191–203, 2014

[268] M. Ying. Floyd-hoare logic for quantum programs. *ACM Trans. Program. Lang. Syst.*, 33(6):19, 2011

[269] M. Ying and Y. Feng. An algebraic language for distributed quantum computing. *IEEE Trans. Computers*, 58(6):728–743, 2009

[270] M. Ying, Y. Feng, R. Duan, and Z. Ji. An algebra of quantum processes. *ACM Trans. Comput. Log.*, 10(3), 2009

[271] M. Ying, Y. Li, N. Yu, and Y. Feng. Model-checking linear-time properties of quantum systems. *ACM Trans. Comput. Log.*, 15(3):22:1–22:31, 2014

[272] M. Ying, N. Yu, Y. Feng, and R. Duan. Verification of quantum programs. *CoRR*, abs/1106.4063, 2011

[273] M. Ying, N. Yu, Y. Feng, and R. Duan. Verification of quantum programs. *Sci. Comput. Program.*, 78(9):1679–1700, 2013

[274] S. Ying and M. Ying. Reachability analysis of quantum markov decision processes. *CoRR*, abs/1406.6146, 2014

[275] N. Yu and M. Ying. Reachability and termination analysis of concurrent quantum programs. *CoRR*, abs/1206.1935, 2012

[276] N. Yu and M. Ying. Reachability and termination analysis of concurrent quantum programs. In M. Koutny and I. Ulidowski, editors, *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2012

Bibliography

[1] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[2] Samson Abramsky. A structural approach to reversible computation. *Theor. Comput. Sci.*, 347(3):441–464, 2005.

[3] Naoki Nishida, Adrián Palacios, and Germán Vidal. Reversible term rewriting in practice. In H. Cirstea and S. Escobar, editors, *Proc. of the Third International Workshop on Rewriting Techniques for Program Transformations and Evaluation (WPTE'16)*, 2016.

[4] Kenichi Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theor. Comput. Sci.*, 148(1):157–163, 1995.

[5] The Quantomatic tool. `quantomatic.github.io`.

[6] S. M. Abramov and R. Glück. The universal resolving algorithm and its correctness: inverse computation in a functional language. *Sci. Comput. Program.*, 43(2-3):193–229, 2002.

[7] S. Abramsky. Retracing some paths in process algebra. In U. Montanari and V. Sassone, editors, *CONCUR '96*, volume 1119 of *Lecture Notes in Computer Science*, pages 1–17. Springer, 1996.

[8] S. Abramsky. A structural approach to reversible computation. *Theoretical Computer Science*, 347(3):441–464, 2005.

[9] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004*, pages 415–425. IEEE, 2004.

[10] S. Abramsky, E. Haghverdi, and P. Scott. Geometry of Interaction and linear combinatory algebras. *Mathematical Structures in Computer Science*, 12(5):625–665, 2002.

[11] J. M. Almendros-Jiménez and G. Vidal. Automatic partial inversion of inductively sequential functions. In *Implementation and Application of Functional Languages, 18th International Symp osium, IFL 2006, Budapest, Hungary, September 4-6, 2006, Revised Selected Papers*, volume 4449 of *Lecture Notes in Computer Science*, pages 253–270. Springer, 2006.

[12] D. Angluin. Inference of reversible languages. *J. ACM*, 29(3):741–765, 1982.

[13] T. Araki and T. Kasami. Decidable problems on the strong connectivity of Petri net reachability sets. *Theoretical Computer Science*, 4(1):99–119, 1977.

[14] E. Ardeshir-Larijani. *Automated Equivalence Checking of Quantum Information Systems*. PhD thesis, University of Warwick, 2014.

[15] E. Ardeshir-Larijani, S. J. Gay, and R. Nagarajan. Automated verification of quantum protocols by equivalence checking. *CoRR*, abs/1312.5951, 2013.

[16] E. Ardeshir-Larijani, S. J. Gay, and R. Nagarajan. Equivalence checking of quantum protocols. In N. Piterman and S. A. Smolka, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 19th International Conference, TACAS 2013, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2013, Rome, Italy, March 16-24, 2013. Proceedings*, volume 7795 of *Lecture Notes in Computer Science*, pages 478–492. Springer, 2013.

[17] E. Ardeshir-Larijani, S. J. Gay, and R. Nagarajan. Verification of concurrent quantum protocols by equivalence checking. In E. Ábrahám and K. Havelund, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*, volume 8413 of *Lecture Notes in Computer Science*, pages 500–514. Springer, 2014.

[18] C. Aubert and I. Cristescu. Reversible barbed congruence on configuration structures. In *ICE*, volume 189 of *EPTCS*, pages 68–85, 2015.

[19] H. B. Axelsen. Reversible multi-head finite automata characterize reversible logarithmic space. In A. H. Dediu and C. Martín-Vide, editors, *Language and Automata Theory and Applications (LATA 2012)*, volume 7183 of *LNCS*, pages 95–105. Springer, 2012.

[20] H. B. Axelsen. Time complexity of tape reduction for reversible Turing machines. In A. D. Vos and R. Wille, editors, *Reversible Computation (RC 2011)*, volume 7165 of *LNCS*, pages 1–13. Springer, 2012.

[21] H. B. Axelsen, S. Jakobi, M. Kutrib, and A. Malcher. A hierarchy of fast reversible Turing machines. In J. Krivine and J.-B. Stefani, editors, *Reversible Computation (RC 2015)*, volume 9138 of *LNCS*, pages 29–44. Springer, 2015.

[22] H. B. Axelsen and R. Kaarsgaard. Join inverse categories as models of reversible recursion. In B. Jacobs and C. Löding, editors, *Foundations of Software Science and Computation Structures: 19th International Conference, FOSSACS 2016, Proceedings*, volume 9634 of *Lecture Notes in Computer Science*, pages 73–90. Springer, 2016.

[23] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.

[24] G. Bacci, V. Danos, and O. Kammar. On the statistical thermodynamics of reversible communicating processes. In *CALCO*, volume 6859 of *LNCS*, pages 1–18. Springer, 2011.

[25] F. Barbanera, M. Dezani-Ciancaglini, and U. de'Liguoro. Compliance for reversible client/server interactions. In *BEAT*, volume 162 of *EPTCS*, pages 35–42, 2014.

[26] F. Barbanera, M. Dezani-Ciancaglini, I. Lanese, and U. de'Liguoro. Retractable contracts. In *PLACES*, volume 203 of *EPTCS*, pages 61–72, 2015.

[27] S. Barbieri, J. Kari, and V. Salo. *The Group of Reversible Turing Machines*, pages 49–62. Springer International Publishing, 2016.

[28] M. Bednarczyk. Hereditary history preserving bisimulations or what is the power of the future perfect in program logics. Technical Report ICS PAS, Polish Academy of Sciences, 1991.

[29] C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Dev.*, 17:525–532, 1973.

[30] G. Berry and G. Boudol. The chemical abstract machine. *Theoretical Computer Science*, 96(1):217–248, 1992.

[31] E. Best, J. Desel, and J. Esparza. Traps characterize home states in free choice systems. *Theoretical Computer Science*, 101(2):161–176, 1992.

[32] E. Best and J. Esparza. Existence of home states in Petri nets is decidable. *Information Processing Letters*, 116(6):423–427, 2016.

[33] E. Best and K. Voss. Free choice systems have home states. *Acta Informatica*, 21(1):89–100, 1984.

[34] W. J. Bowman, R. P. James, and A. Sabry. Dagger traced symmetric monoidal categories and reversible programming. In A. De Vos and R. Wille, editors, *Proceedings of RC 2011*, pages 51–56. Ghent University, 2011.

[35] M. Boyle, D. Lind, and D. Rudolph. The automorphism group of a shift of finite type. *Transactions of the American Mathematical Society*, 306(1):pp. 71–114, 1988.

[36] M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner. *Model-Based Testing of Reactive Systems: Advanced Lectures*, volume 3472 of *Lecture Notes in Computer Science*. Springer, 2005.

[37] R. Bruni, H. C. Melgratti, and U. Montanari. Theoretical foundations for compensations in flow composition languages. In *POPL*, pages 209–220. ACM, 2005.

[38] H. Buhrman, J. Tromp, and P. M. B. Vitányi. Time and space bounds for reversible simulation. In F. Orejas, P. G. Spirakis, and J. van Leeuwen, editors, *International Colloquium on Automata, Languages and Programming (ICALP 2001)*, volume 2076 of *LNCS*, pages 1017–1027. Springer, 2001.

[39] A. W. Burks, H. H. Goldstine, and J. von Neumann. Preliminary discussion of the logical design of an electronic computing instrument. Technical report, Institute of Advanced Study, U.S. Army, 1947.

[40] L. Cardelli and C. Laneve. Reversible structures. In *CMSB*, pages 131–140. ACM, 2011.

[41] J. Carette and A. Sabry. Computing with semirings and weak rig groupoids. In P. Thiemann, editor, *Programming Languages and Systems: 25th European Symposium on Programming, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 123–148. Springer, 2016.

[42] K. Chatterjee and J. Sgall, editors. *Mathematical Foundations of Computer Science 2013 - 38th International Symposium, MFCS 2013, Klosterneuburg, Austria, August 26-30, 2013. Proceedings*, volume 8087 of *Lecture Notes in Computer Science*. Springer, 2013.

[43] T. S. Chow. Testing Software Design Modeled by Finite-State Machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, May 1978.

[44] J. R. B. Cockett and S. Lack. Restriction categories I: Categories of partial maps. *Theoretical Computer Science*, 270(1–2):223–259, 2002.

[45] I. Cristescu, J. Krivine, and D. Varacca. A compositional semantics for the reversible pi-calculus. In *LICS*, pages 388–397. IEEE Computer Society, 2013.

[46] E. Czeizler and J. Kari. A tight linear bound on the neighborhood of inverse cellular automata. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 410–420, 2005.

[47] N. P. Danko Kezic and I. Petrovic. An algorithm for deadlock prevention based on iterative siphon control of Petri net. *Automatika*, 47(1-2):19–30, 2006.

[48] V. Danos and J. Krivine. Formal molecular biology done in CCS-R. In *BioConcur*, volume 180(3) of *ENTCS*, pages 31–49. Elsevier, 2003.

[49] V. Danos and J. Krivine. Reversible communicating systems. In *CONCUR*, volume 3170 of *LNCS*, pages 292–307. Springer, 2004.

[50] V. Danos and J. Krivine. Transactions in RCCS. In *CONCUR*, volume 3653 of *LNCS*, pages 398–412. Springer, 2005.

[51] V. Danos, J. Krivine, and P. Sobocinski. General reversibility. In *SOS*, volume 175(3) of *ENTCS*, pages 75–86. Elsevier, 2006.

[52] V. Danos, J. Krivine, and F. Tarissan. Self-assembling trees. In *SOS*, volume 175(1) of *ENTCS*, pages 19–32. Elsevier, 2007.

[53] T. A. S. Davidson. *Formal Verification Techniques using Quantum Process Calculus*. PhD thesis, University of Warwick, 2012.

[54] T. A. S. Davidson, S. J. Gay, H. Mlnarik, R. Nagarajan, and N. Papanikolaou. Model checking for communicating quantum processes. *IJUC*, 8(1):73–98, 2012.

[55] T. A. S. Davidson, S. J. Gay, and R. Nagarajan. Formal analysis of quantum systems using process calculus. In A. Silva, S. Bliudze, R. Bruni, and M. Carbone, editors, *Proceedings Fourth Interaction and Concurrency Experience, ICE 2011, Reykjavik, Iceland, 9th June 2011.*, volume 59 of *EPTCS*, pages 104–110, 2011.

[56] T. A. S. Davidson, S. J. Gay, R. Nagarajan, and I. V. Puthoor. Analysis of a quantum error correcting code using quantum process calculus. In B. Jacobs, P. Selinger, and B. Spitters, editors, *Proceedings 8th International Workshop on Quantum Physics and Logic, QPL 2011, Nijmegen, Netherlands, October 27-29, 2011.*, volume 95 of *EPTCS*, pages 67–80, 2011.

[57] D. de Frutos Escrig. Decidability of home states in place transition systems. Internal report. dpto. informatica y automatica, Univ. Complutense de Madrid, 1986.

[58] D. de Frutos Escrig and C. Johnen. Decidability of home space property. Report LRI 503, Univ. de Paris-Sud, 1989.

[59] E. de Vries, V. Koutavas, and M. Hennessy. Communicating transactions - (extended abstract). In *CONCUR*, volume 6269 of *Lecture Notes in Computer Science*, pages 569–583. Springer, 2010.

[60] E. de Vries, V. Koutavas, and M. Hennessy. Liveness of communicating transactions (extended abstract). In *APLAS*, volume 6461 of *LNCS*, pages 392–407. Springer, 2010.

[61] Y. Deng and Y. Feng. Open bisimulation for quantum processes. *CoRR*, abs/1201.0416, 2012.

[62] Y. Deng and Y. Feng. Open bisimulation for quantum processes. In J. C. M. Baeten, T. Ball, and F. S. de Boer, editors, *Theoretical Computer Science - 7th IFIP TC 1/WG 2.2 International Conference, TCS 2012, Amsterdam, The Netherlands, September 26-28, 2012. Proceedings*, volume 7604 of *Lecture Notes in Computer Science*, pages 119–133. Springer, 2012.

[63] N. Dershowitz and S. Mitra. Jeopardy. In *RTA*, pages 16–29, 1999.

[64] J. Desel and J. Esparza. Reachability in cyclic extended free-choice systems. *Theoretical Computer Science*, 114(1):93–118, 1993.

[65] L. Dixon and R. Duncan. Extending graphical representations for compact closed categories with applications to symbolic quantum computation. In S. Autexier, J. A. Campbell, J. Rubio, V. Sorge, M. Suzuki, and F. Wiedijk, editors, *Intelligent Computer Mathematics, 9th International Conference, AISC 2008, 15th Symposium, Calculemus 2008, 7th International Conference, MKM 2008, Birmingham, UK, July 28 - August 1, 2008. Proceedings*, volume 5144 of *Lecture Notes in Computer Science*, pages 77–92. Springer, 2008.

[66] L. Dixon and R. Duncan. Graphical reasoning in compact closed categories for quantum computation. *Ann. Math. Artif. Intell.*, 56(1):23–42, 2009.

[67] L. Dixon, R. Duncan, and A. Kissinger. Open graphs and computational reasoning. In S. B. Cooper, P. Panangaden, and E. Kashefi, editors, *Proceedings Sixth Workshop on Developments in Computational Models: Causality, Computation, and Physics, DCM 2010, Edinburgh, Scotland, 9-10th July 2010.*, volume 26 of *EPTCS*, pages 169–180, 2010.

[68] L. Dixon and A. Kissinger. Open graphs and monoidal theories. *CoRR*, abs/1011.4114, 2010.

[69] J.-C. Dubacq. How to simulate Turing machines by invertible one-dimensional cellular automata. *International Journal of Foundations of Computer Science*, 6(4):395–402, 1995.

[70] R. Duncan and M. Lucas. Verifying the steane code with quantomatic. In B. Coecke and M. J. Hoban, editors, *Proceedings of the 10th International Workshop on Quantum Physics and Logic, QPL 2013, Castelldefels (Barcelona), Spain, July 17-19, 2013.*, volume 171 of *EPTCS*, pages 33–49, 2013.

[71] J. O. Durand-Lose. Reversible space-time simulation of cellular automata. *Theoretical Computer Science*, 246:117–129, 2000.

[72] J. O. Durand-Lose. Representing reversible cellular automata with reversible block cellular automata. In *Discrete Models: Combinatorics, Computation, and Geometry, DM-CCG 2001, Proceedings*, pages 145–154, 2001.

[73] Y. Feng, Y. Deng, and M. Ying. Symbolic bisimulation for quantum processes. *CoRR*, abs/1202.3484, 2012.

[74] Y. Feng, Y. Deng, and M. Ying. Symbolic bisimulation for quantum processes. *ACM Trans. Comput. Log.*, 15(2):14, 2014.

[75] Y. Feng, R. Duan, Z. Ji, and M. Ying. Probabilistic bisimilarities between quantum processes. *CoRR*, abs/cs/0601014, 2006.

[76] Y. Feng, R. Duan, Z. Ji, and M. Ying. Probabilistic bisimulations for quantum processes. *Inf. Comput.*, 205(11):1608–1639, 2007.

[77] Y. Feng, R. Duan, and M. Ying. Bisimulation for quantum processes. *CoRR*, abs/1007.2584, 2010.

[78] Y. Feng, R. Duan, and M. Ying. Bisimulation for quantum processes. In T. Ball and M. Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2011, Austin, TX, USA, January 26-28, 2011*, pages 523–534. ACM, 2011.

[79] Y. Feng, R. Duan, and M. Ying. Bisimulation for quantum processes. *ACM Trans. Program. Lang. Syst.*, 34(4):17, 2012.

[80] Y. Feng, E. M. Hahn, A. Turrini, and L. Zhang. QPMC: A model checker for quantum programs and protocols. In N. Bjørner and F. S. de Boer, editors, *FM 2015: Formal Methods - 20th International Symposium, Oslo, Norway, June 24-26, 2015, Proceedings*, volume 9109 of *Lecture Notes in Computer Science*, pages 265–272. Springer, 2015.

[81] Y. Feng and M. Ying. Toward automatic verification of quantum cryptographic protocols. *CoRR*, abs/1507.05278, 2015.

[82] Y. Feng and M. Ying. Toward automatic verification of quantum cryptographic protocols. In L. Aceto and D. de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 441–455. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[83] Y. Feng, N. Yu, and M. Ying. Model checking quantum markov chains. *CoRR*, abs/1205.2187, 2012.

[84] Y. Feng, N. Yu, and M. Ying. Model checking quantum markov chains. *J. Comput. Syst. Sci.*, 79(7):1181–1198, 2013.

[85] Y. Feng, N. Yu, and M. Ying. Reachability analysis of recursive quantum markov chains. In Chatterjee and Sgall [42], pages 385–396.

[86] L. Ferrarini. On the reachability and reversibility problems in a class of Petri nets. *IEEE Transactions on Systems, Man and Cybernetics*, 24(10):1474–1482, 1994.

[87] D. Fiebig and U.-R. Fiebig. The automorphism group of a coded system. *Transactions of the American Mathematical Society*, 348(8):3173–3191, 1996.

[88] N. Foster, K. Matsuda, and J. Voigtländer. Three complementary approaches to bidirectional programming. In J. Gibbons, editor, *Generic and Indexed Programming - International Spring School, SSGIP 2010, Oxford, UK, March 22-26, 2010, Revised Lectures*, volume 7470 of *Lecture Notes in Computer Science*, pages 1–46. Springer, 2012.

[89] S. Franke-Arnold, S. J. Gay, and I. V. Puthoor. Quantum process calculus for linear optical quantum computing. In G. W. Dueck and D. M. Miller, editors, *Reversible Computation - 5th International Conference, RC 2013, Victoria, BC, Canada, July 4-5, 2013. Proceedings*, volume 7948 of *Lecture Notes in Computer Science*, pages 234–246. Springer, 2013.

[90] S. Franke-Arnold, S. J. Gay, and I. V. Puthoor. Verification of linear optical quantum computing using quantum process calculus. In J. Borgström and S. Crafa, editors, *Proceedings Combined 21st International Workshop on Expressiveness in Concurrency and 11th Workshop on Structural Operational Semantics, EXPRESS 2014, and 11th Workshop on Structural Operational Semantics, SOS 2014, Rome, Italy, 1st September 2014.*, volume 160 of *EPTCS*, pages 111–129, 2014.

[91] A. Gajardo, J. Kari, and A. Moreira. On time-symmetry in cellular automata. *J. Comput. System Sci.*, 78:1115–1126, 2012.

[92] S. J. Gay and R. Nagarajan. Communicating quantum processes. In J. Palsberg and M. Abadi, editors, *Proceedings of the 32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005, Long Beach, California, USA, January 12-14, 2005*, pages 145–157. ACM, 2005.

[93] S. J. Gay and R. Nagarajan. Types and typechecking for communicating quantum processes. *Mathematical Structures in Computer Science*, 16(3):375–406, 2006.

[94] S. J. Gay and R. Nagarajan. Techniques for formal modelling and analysis of quantum systems. In B. Coecke, L. Ong, and P. Panangaden, editors, *Computation, Logic, Games, and Quantum Foundations. The Many Facets of Samson Abramsky - Essays Dedicated to Samson Abramsky on the Occasion of His 60th Birthday*, volume 7860 of *Lecture Notes in Computer Science*, pages 264–276. Springer, 2013.

[95] S. J. Gay, R. Nagarajan, and N. Papanikolaou. Probabilistic model-checking of quantum protocols. *CoRR*, abs/quant-ph/0504007, 2005.

[96] S. J. Gay, R. Nagarajan, and N. Papanikolaou. QMC: A model checker for quantum systems. In A. Gupta and S. Malik, editors, *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*, volume 5123 of *Lecture Notes in Computer Science*, pages 543–547. Springer, 2008.

[97] S. J. Gay, R. Nagarajan, and N. Papanikolaou. Specification and verification of quantum protocols. In S. J. Gay and I. C. Mackie, editors, *Semantic Techniques in Quantum Computation*. Cambridge University Press, 2010.

[98] S. J. Gay and I. V. Puthoor. Application of quantum process calculus to higher dimensional quantum protocols. In R. Duncan and P. Panangaden, editors, *Proceedings 9th Workshop on Quantum Physics and Logic, QPL 2012, Brussels, Belgium, 10-12 October 2012.*, volume 158 of *EPTCS*, pages 15–28, 2014.

[99] S. J. Gay and I. V. Puthoor. Equational reasoning about quantum protocols. In J. Krivine and J. Stefani, editors, *Reversible Computation - 7th International Conference, RC 2015, Grenoble, France, July 16-17, 2015, Proceedings*, volume 9138 of *Lecture Notes in Computer Science*, pages 155–170. Springer, 2015.

[100] E. Giachino, I. Lanese, C. A. Mezzina, and F. Tiezzi. Causal-consistent reversibility in a tuple-based language. In *PDP*, pages 467–475. IEEE Computer Society, 2015.

[101] B. G. Giles. *An investigation of some theoretical aspects of reversible computing*. PhD thesis, University of Calgary, 2014.

[102] R. v. Glabbeek and U. Goltz. Refinement of actions and equivalence notions for concurrent systems. *Acta Informatica*, 37:229–327, 2001.

[103] R. Glück and M. Kawabe. A program inverter for a functional language with equality and constructors. In *Proceedings of the first Asian Symposium on Programming Languages and Systems (APLAS 2003)*, pages 246–264, 2003.

[104] R. Glück and M. Kawabe. A method for automatic program inversion based on LR(0) parsing. *Fundam. Inform.*, 66(4):367–395, 2005.

[105] M. Goudarzi, J. Chen, D. Vasudevan, E. Popovici, and M. Schellekens. Reversing deterministic finite state machines. In *Signals and Systems Conference (ISSC 2009), IET Irish*, pages 1–6, June 2009.

[106] C. Güniçen, K. Inan, U. C. Türker, and H. Yenigün. The relation between preset distinguishing sequences and synchronizing sequences. *Formal Asp. Comput.*, 26(6):1153–1167, 2014.

[107] X. Guo. *Products, Joins, Meets, and Ranges in Restriction Categories.* PhD thesis, University of Calgary, 2012.

[108] M. Hack. *Decidability Questions for Petri Nets.* PhD thesis, MIT, 1975.

[109] P. G. Harrison. Function inversion. In D. Bjørner, A. P. Ershov, and N. D. Jones, editors, *Proceedings of the International Workshop on Partial Evaluation and Mixed Computation*, pages 153–166. North-Holland, Amsterdam, 1988.

[110] G. A. Hedlund. Endomorphisms and automorphisms of the shift dynamical system. *Mathematical Systems Theory*, 3:320–375, 1969.

[111] M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In *ICALP*, volume 85 of *LNCS*, pages 299–309. Springer, 1980.

[112] P. Hertling. Embedding cellular automata into reversible ones. In C. S. Calude, J. Casti, and M. J. Dinneen, editors, *Unconventional Models of Computation, Proceedings*, pages 243–256. Springer, 1998.

[113] C. Heunen and M. Karvonen. Reversible monadic computing. *Electronic Notes in Theoretical Computer Science*, 319:217–237, 2015.

[114] C. Heunen and M. Karvonen. Monads on dagger categories. arXiv preprint, arXiv:1602.04324, 2016.

[115] R. M. Hierons. Extending test sequence overlap by invertibility. *Comput. J.*, 39(4):325–330, 1996.

[116] R. M. Hierons. Testing from a finite-state machine: Extending invertibility to sequences. *Comput. J.*, 40(4):220–230, 1997.

[117] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Lüttgen, A. J. H. Simons, S. Vilkomir, M. R. Woodward, and H. Zedan. Using Formal Specifications to Support Testing. *ACM Computing Surveys*, 41(2):9:1–9:76, 2009.

[118] R. M. Hierons, M. R. Mousavi, M. Kirkedal Thomsen, and U. C. Türker. Hardness of deriving invertible sequences from finite state machines. Submitted.

[119] P. M. Hines. *The Algebra of Self-Similarity and its Applications*. PhD thesis, University of Wales, Bangor, 1998.

[120] M. Hochman. On the automorphism groups of multidimensional shifts of finite type. *Ergodic Theory Dynam. Systems*, 30(3):809–840, 2010.

[121] M. Holzer, S. Jakobi, and M. Kutrib. Minimal reversible deterministic finite automata. In I. Potapov, editor, *Developments in Language Theory (DLT 2015)*, volume 9168 of *LNCS*, pages 276–287. Springer, 2015.

[122] D. A. Huffman. Canonical forms for information-lossless finite-state logical machines. *IRE Transactions on Information Theory*, 5(5):41–59, 1959.

[123] T. Hujsa, J.-M. Delosme, and A. M. Kordon. On the reversibility of live equal-conflict Petri nets. In R. R. Devillers and A. Valmari, editors, *Application and Theory of Petri Nets and Concurrency - 36th International Conference, PETRI NETS 2015, Brussels, Belgium, June 21-26, 2015, Proceedings*, volume 9115 of *Lecture Notes in Computer Science*, pages 234–253. Springer, 2015.

[124] H. Hüttel, I. Lanese, V. T. Vasconcelos, L. Caires, M. Carbone, P.-M. Deniélou, D. Mostrous, L. Padovani, A. Ravara, E. Tuosto, H. T. Vieira, and G. Zavattaro. Foundations of session types and behavioural contracts. *ACM Comput. Surv.*, 49(1):3:1–3:36, 2016.

[125] B. Jacobs. New directions in categorical logic, for classical, probabilistic and quantum logic. *Logical Methods in Computer Science*, 11(3):1–76, 2015.

[126] R. P. James and A. Sabry. Information effects. *ACM SIGPLAN Notices*, 47(1):73–84, 2012.

[127] P. Jorrand and M. Lalire. From quantum physics to programming languages: A process algebraic approach. In J. Banâtre, P. Fradet, J. Giavitto, and O. Michel, editors, *Unconventional Programming Paradigms, International Workshop UPP 2004, Le Mont Saint Michel, France, September 15-17, 2004, Revised Selected and Invited Papers*, volume 3566 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2004.

[128] P. Jorrand and M. Lalire. Toward a quantum process algebra. In S. Vassiliadis, J. Gaudiot, and V. Piuri, editors, *Proceedings of the First Conference on Computing Frontiers, 2004, Ischia, Italy, April 14-16, 2004*, pages 111–119. ACM, 2004.

[129] A. Joyal, R. Street, and D. Verity. Traced monoidal categories. *Mathematical Proceedings of the Cambridge Philosophical Society*, 119(3):447–468, 1996.

[130] J. Kari. Reversibility and surjectivity problems of cellular automata. *Journal of Computer and System Sciences*, 48(1):149–182, Feb. 1994.

[131] J. Kari. Representation of reversible cellular automata with block permutations. *Mathematical Systems Theory*, 29(1):47–61, 1996.

[132] J. Kari and N. Ollinger. Periodicity and immortality in reversible computing. In *Mathematical Foundations of Computer Science 2008, 33rd International Symposium, MFCS 2008, Torun, Poland, August 25-29, 2008, Proceedings*, pages 419–430, 2008.

[133] M. Kawabe and Y. Futamura. Case studies with an automatic program inversion system. In *Proceedings of the 21st Conference of Japan Society for Software Science and Technology*, number 6C-3, 5 pages, 2004.

[134] M. Kawabe and R. Glück. The program inverter LRinv and its structure. In *Proceedings of the 7th International Symposium on Practical Aspects of Declarative Languages (PADL 2005)*, pages 219–234, 2005.

[135] R. Keller. Towards a theory of universal speed-independent modules. *IEEE Transactions on Computers*, 23(1):21–33, 1974.

[136] H. Khoshnevisan and K. M. Sephton. InvX: An automatic function inverter. In N. Dershowitz, editor, *Proceedings of the 3rd International Conference of Rewriting Techniques and Applications (RTA'89)*, volume 355 of *Lecture Notes in Computer Science*, pages 564–568. Springer, 1989.

[137] K. H. Kim and F. W. Roush. On the automorphism groups of subshifts. *Pure Mathematics and Applications*, 1(4):203–230, 1990.

[138] A. Kissinger. Exploring a quantum theory with graph rewriting and computer algebra. In J. Carette, L. Dixon, C. S. Coen, and S. M. Watt, editors, *Intelligent Computer Mathematics, 16th Symposium, Calculemus 2009, 8th International Conference, MKM 2009, Held as Part of CICM 2009, Grand Bend, Canada, July 6-12, 2009. Proceedings*, volume 5625 of *Lecture Notes in Computer Science*, pages 90–105. Springer, 2009.

[139] A. Kissinger. Pictures of processes: Automated graph rewriting for monoidal categories and applications to quantum computing. *CoRR*, abs/1203.0202, 2012.

[140] A. Kissinger and V. Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. *CoRR*, abs/1503.01034, 2015.

[141] A. Kissinger and V. Zamdzhiev. Quantomatic: A proof assistant for diagrammatic reasoning. In A. P. Felty and A. Middeldorp, editors, *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecture Notes in Computer Science*, pages 326–336. Springer, 2015.

[142] J. W. Klop. Term Rewriting Systems. In S. Abramsky, D. Gabbay, and T. Maibaum, editors, *Handbook of Logic in Computer Science*, volume I, pages 1–112. Oxford University Press, 1992.

[143] A. Kondacs and J. Watrous. On the power of quantum finite state automata. In *Foundations of Computer Science (FOCS 1997)*, pages 66–75. IEEE Computer Society, 1997.

[144] V. Koutavas, C. Spaccasassi, and M. Hennessy. Bisimulations for communicating transactions - (extended abstract). In *FOSSACS*, volume 8412 of *LNCS*, pages 320–334. Springer, 2014.

[145] J. Krivine. A verification technique for reversible process algebra. In *RC*, volume 7581 of *LNCS*, pages 204–217. Springer, 2012.

[146] T. Kubota, Y. Kakutani, G. Kato, Y. Kawano, and H. Sakurada. Automated verification of equivalence on quantum cryptographic protocols. In L. Kovács and T. Kutsia, editors, *5th International Symposium on Symbolic Computation in Software Science, SCSS 2013*, volume 15 of *EPiC Series*, pages 64–69. EasyChair, 2013.

[147] T. Kubota, Y. Kakutani, G. Kato, Y. Kawano, and H. Sakurada. Semi-automated verification of security proofs of quantum cryptographic protocols. *J. Symb. Comput.*, 73:192–220, 2016.

[148] S. Kuhn and I. Ulidowski. Towards modelling of local reversibility. In *RC*, volume 9138 of *LNCS*, pages 279–284. Springer, 2015.

[149] S. Kuhn and I. Ulidowski. A calculus for local reversibility. In *RC*, volume 9720 of *LNCS*, pages 20–35. Springer, 2016.

[150] M. Kutrib. Reversible and irreversible computations of deterministic finite-state devices. In G. F. Italiano, G. Pighizzini, and D. Sannella, editors, *Mathematical Foundations of Computer Science (MFCS 2015)*, volume 9234 of *LNCS*, pages 38–52. Springer, 2015.

[151] M. Kutrib and A. Malcher. Reversible pushdown automata. *J. Comput. System Sci.*, 78:1814–1827, 2012.

[152] M. Kutrib and A. Malcher. One-way reversible multi-head finite automata. In R. Glück and T. Yokoyama, editors, *Reversible Computation (RC 2012)*, volume 7581 of *LNCS*, pages 14–28. Springer, 2013.

[153] M. Kutrib, A. Malcher, and M. Wendlandt. Reversible queue automata. In *Non-Classical Models of Automata and Applications (NCMA 2014)*, volume 304 of *books@ocg.at*, pages 163–178, Vienna, 2014. Austrian Computer Society.

[154] M. Kutrib, A. Malcher, and M. Wendlandt. When input-driven pushdown automata meet reversibility. In R. Freund, M. Holzer, N. Moreira, and R. Reis, editors, *Non-Classical Models of Automata and Applications (NCMA 2015)*, volume 318 of *books@ocg.at*, pages 141–157, Vienna, 2015. Austrian Computer Society.

[155] M. Kutrib and M. Wendlandt. Reversible limited automata. In J. Durand-Lose and B. Nagy, editors, *Machines, Computations, and Universality (MCU 2015)*, volume 9288 of *LNCS*, pages 113–128. Springer, 2015.

[156] M. Kutrib and T. Worsch. Time-symmetric machines. In G. W. Dueck and D. M. Miller, editors, *Reversible Computation (RC 2013)*, volume 7948 of *LNCS*, pages 168–181. Springer, 2013.

[157] M. Kutrib and T. Worsch. Degrees of reversibility for DFA and DPDA. In Y. Shigeru and M. Shin-ichi, editors, *Reversible Computation (RC 2014)*, volume 8507 of *LNCS*, pages 40–53. Springer, 2014.

[158] M. Lalire. A probabilistic branching bisimulation for quantum processes. *CoRR*, abs/quant-ph/0508116, 2005.

[159] M. Lalire. Relations among quantum processes: bisimilarity and congruence. *Mathematical Structures in Computer Science*, 16(3):407–428, 2006.

[160] M. Lalire and P. Jorrand. A process algebraic approach to concurrent and distributed quantum computation: Operational semantics. *CoRR*, quant-ph/0407005, 2004.

[161] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 5(3):183–191, 1961.

[162] R. Landauer. Information is physical. *Physics Today*, 44(5):23–29, 1991.

[163] R. Landauer. Zig-zag path to understanding. In *Workshop on Physics and Computation, PhysComp '94, Proceedings*, pages 54–59. IEEE, 1994.

[164] I. Lanese, M. Lienhardt, C. A. Mezzina, A. Schmitt, and J.-B. Stefani. Concurrent flexible reversibility. In *ESOP*, volume 7792 of *LNCS*, pages 370–390. Springer, 2013.

[165] I. Lanese, C. A. Mezzina, A. Schmitt, and J.-B. Stefani. Controlling reversibility in higher-order pi. In *CONCUR*, volume 6901 of *LNCS*, pages 297–311, 2011.

[166] I. Lanese, C. A. Mezzina, and J. Stefani. Reversibility in the higher-order π-calculus. *Theor. Comput. Sci.*, 625:25–84, 2016.

[167] I. Lanese, C. A. Mezzina, and J.-B. Stefani. Reversing higher-order pi. In *CONCUR*, volume 6269 of *LNCS*, pages 478–493. Springer, 2010.

[168] I. Lanese, C. A. Mezzina, and J.-B. Stefani. Controlled reversibility and compensations. In *RC*, volume 7581 of *LNCS*, pages 233–240. Springer, 2012.

[169] I. Lanese, C. A. Mezzina, and F. Tiezzi. Causal-consistent reversibility. *Bulletin of the EATCS*, 114:121–139, 2014.

[170] K.-J. Lange, P. McKenzie, and A. Tapp. Reversible space equals deterministic space. *J. Comput. System Sci.*, 60:354–367, 2000.

[171] Y. Lecerf. Machines de Turing réversibles. *Comptes Rendus Hebdomadaires des Séances de l'Académie des Sciences*, 257:2597–2600, 1963.

[172] J. Lee, S. Adachi, F. Peper, and K. Morita. Embedding universal delay-insensitive circuits in asynchronous cellular spaces. *Fundamenta Informaticae*, 58(3-4):295–320, 2003.

[173] J. Lee, S. Adachi, Y.-N. Xia, and Q.-S. Zhu. Emergence of universal global behavior from reversible local transitions in asynchronous systems. *Information Sciences*, 282:38 – 56, 2014.

[174] J. Lee, X. Huang, and Q.-s. Zhu. Embedding simple reversed-twin elements into self-timed reversible cellular automata. *Journal of Convergence Information Technology*, 6(1), 2011.

[175] J. Lee, F. Peper, S. Adachi, and K. Morita. An asynchronous cellular automaton implementing 2-state 2-input 2-output reversed-twin reversible elements. In *Proceedings of ACRI 2008*, volume 5191 of *LNCS*, pages 67–76. Springer, 2008.

[176] J. Lee, F. Peper, S. Adachi, K. Morita, and S. Mashiko. Reversible computation in asynchronous cellular automata. In *Proceedings of Unconventional Models of Computation, UMC 2002*, pages 220–229, 2002.

[177] L. Li and Y. Feng. Quantum markov chains: description of hybrid systems, decidability of equivalence, and model checking linear-time properties. *CoRR*, abs/1506.08982, 2015.

[178] L. Li and Y. Feng. Quantum markov chains: Description of hybrid systems, decidability of equivalence, and model checking linear-time properties. *Inf. Comput.*, 244:229–244, 2015.

[179] Y. Li and M. Ying. Debugging quantum processes using monitoring measurements. *CoRR*, abs/1403.4344, 2014.

[180] Y. Li and M. Ying. (un)decidable problems about reachability of quantum systems. *CoRR*, abs/1401.6249, 2014.

[181] Y. Li and M. Ying. (un)decidable problems about reachability of quantum systems. In P. Baldan and D. Gorla, editors, *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, volume 8704 of *Lecture Notes in Computer Science*, pages 482–496. Springer, 2014.

[182] Y. Li, N. Yu, and M. Ying. Termination of nondeterministic quantum programs. *CoRR*, abs/1201.0891, 2012.

[183] Y. Li, N. Yu, and M. Ying. Termination of nondeterministic quantum programs. *Acta Inf.*, 51(1):1–24, 2014.

[184] T. Liu, Y. Li, S. Wang, M. Ying, and N. Zhan. A theorem prover for quantum hoare logic and its applications. *CoRR*, abs/1601.03835, 2016.

[185] M. Lukac, M. Kameyama, M. Perkowski, and P. Kerntopf. Analysis of reversible and quantum finite state machines using homing, synchronizing and distinguishing input sequences. In *Multiple-Valued Logic (ISMVL), 2013 IEEE 43rd International Symposium on*, pages 322–327, May 2013.

[186] M. Marchiori. Unraveling and Ultraproperties. In M. Rodríguez-Artalejo and M. Hanus, editors, *Proc. of Sixth Int'l Conf. on Algebraic and Logic Programming, ALP'96*, pages 107–121. Springer LNCS 1139, 1996.

[187] A. J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In *Procs. of AUSCRIPT '90*, pages 263–278. MIT Press, 1990.

[188] K. Matsuda, Z. Hu, K. Nakano, M. Hamana, and M. Takeichi. Bidirectionalization transformation based on automatic derivation of view complement functions. In R. Hinze and N. Ramsey, editors, *Proc. of the 12th ACM SIGPLAN International Conference on Functional Programming, ICFP 2007*, pages 47–58. ACM, 2007.

[189] K. Matsuda, S.-C. Mu, Z. Hu, and M. Takeichi. A grammar-based approach to invertible programs. In *ESOP*, pages 448–467, 2010.

[190] J. McCarthy. The inversion of functions defined by Turing machines. *Automata Studies*, pages 177–181, 1956.

[191] J. D. McGregor and D. A. Sykes. *A Practical Guide to Testing Object-oriented Software*. Object Technology Series. Addison Wesley, 2001.

[192] D. Medic and C. A. Mezzina. Static VS dynamic reversibility in CCS. In *RC*, volume 9720 of *LNCS*, pages 36–51. Springer, 2016.

[193] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[194] T. Æ. Mogensen. Semi-inversion of guarded equations. In R. Glück and M. R. Lowry, editors, *Generative Programming and Component Engineering, 4th International Conference, GPCE 2005, Tallinn, Estonia, Proceedings*, volume 3676 of *Lecture Notes in Computer Science*, pages 189–204. Springer, 2005.

[195] T. Æ. Mogensen. Report on an implementation of a semi-inverter. In I. Virbitskaite and A. Voronkov, editors, *Perspectives of Systems Informatics, 6th International Andrei Ershov Memorial Conference, PSI 2006. Revised Papers*, volume 4378 of *Lecture Notes in Computer Science*, pages 322–334. Springer, 2007.

[196] T. Æ. Mogensen. Semi-inversion of functional parameters. In R. Glück and O. de Moor, editors, *Proceedings of the 2008 ACM SIGPLAN Symposium on Partial Evaluation and Semantics-based Program Manipulation, PEPM 2008, San Francisco, California, USA*, pages 21–29. ACM, 2008.

[197] E. F. Moore. Machine models of self-reproduction. *Proceedings Symposium Applied Mathematics*, 14:17–33, 1963.

[198] K. Morita. Reversible simulation of one-dimensional irreversible cellular automata. *Theoretical Computer Science*, 148(1):157–163, 1995.

[199] K. Morita. A simple universal logic element and cellular automata for reversible computing. In *Proceedings of MCU 2001*, volume 2055 of *LNCS*, pages 102–113. Springer, 2001.

[200] K. Morita. Two-way reversible multi-head finite automata. *Fund. Inform.*, 110:241–254, 2011.

[201] K. Morita. Reversible computing systems, logic circuits, and cellular automata. In *Proceedings of Networking and Computing 2012*, pages 1–8, 2012.

[202] K. Morita. Universality of one-dimensional reversible and number-conserving cellular automata. In *Proceedings 18th international workshop on Cellular Automata and Discrete Complex Systems and 3rd international symposium Journées Automates Cellulaires, AUTOMATA & JAC 2012*, pages 142–150, 2012.

[203] K. Morita and M. Harao. Computation universality of one-dimensional reversible (injective) cellular automata. *IEICE Transactions (1976-1990)*, 72:758–762, 1989.

[204] K. Morita, T. Ogiro, K. Tanaka, and H. Kato. Classification and universality of reversible logic elements with one-bit memory. In *Proceedings of MCU 2004*, volume 3354 of *LNCS*, pages 245–256. Springer, 2004.

[205] D. Morrison and I. Ulidowski. Reversible delay-insensitive distributed memory modules. In *Proceedings of Reversible Computation 2013*, volume 7948 of *LNCS*, pages 11–24. Springer, 2013.

[206] D. Morrison and I. Ulidowski. Arbitration and reversibility of parallel delay-insensitive modules. In *Proceedings of Reversible Computation 2014*, volume 8507 of *LNCS*, pages 67–81. Springer, 2014.

[207] D. Morrison and I. Ulidowski. Direction-reversible self-timed cellular automata for delay-insensitive circuits. In *Proceedings of ACRI 2013*, volume 8751 of *LNCS*, pages 367–377. Springer, 2014.

[208] T. Murata. Petri nets: properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[209] J. Myhill. The converse to Moore's garden-of-eden theorem. *Proceedings of the American Mathematical Society*, 14:685–686, 1963.

[210] R. Nagarajan and S. J. Gay. Formal verification of quantum protocols. *arXiv*, quant-ph/0203086,, 2002.

[211] R. Nagarajan, N. Papanikolaou, G. Bowen, and S. J. Gay. An automated analysis of the security of quantum key distribution. *CoRR*, abs/cs/0502048, 2005.

[212] N. Nishida. *Transformational Approach to Inverse Computation in Term Rewriting*. PhD thesis, Graduate School of Engineering, Nagoya University, Nagoya, Japan, Jan 2004.

[213] N. Nishida, A. Palacios, and G. Vidal. Reversible term rewriting. In D. Kesner and B. Pientka, editors, *1st International Conference on Formal Structures for Computation and Deduction, FSCD 2016, June 22-26, 2016, Porto, Portugal*, volume 52 of *LIPIcs*, pages 28:1–28:18. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[214] N. Nishida and M. Sakai. Completion after program inversion of injective functions. *Electr. Notes Theor. Comput. Sci.*, 237:39–56, 2009.

[215] N. Nishida, M. Sakai, and T. Sakabe. Generation of inverse term rewriting systems for pure treeless functions. In Y. Toyama, editor, *Proceedings of the International Workshop on Rewriting in Proof and Computation (RPC'01), Sendai, Japan*, pages 188–198, Oct 2001.

[216] N. Nishida, M. Sakai, and T. Sakabe. Partial inversion of constructor term rewriting systems. In *Proceedings of the 16th International Conference on Rewriting Techniques and Applications (RTA 2005)*, pages 264–278, 2005.

[217] N. Nishida and G. Vidal. Program inversion for tail recursive functions. In *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications (RTA 2011)*, pages 283–298, 2011.

[218] E. Ohlebusch. *Advanced topics in term rewriting*. Springer, 2002.

[219] N. Papanikolaou. Techniques for design and validation of quantum protocols. Master's thesis, University of Warwick, 2004.

[220] N. Papanikolaou. *Model Checking Quantum Protocols*. PhD thesis, University of Warwick, 2009.

[221] F. Peper, T. Isokawa, N. Kouda, and N. Matsui. Self-timed cellular automata and their computational ability. *Future Generation Computer Systems*, 18(7):893–904, 2002.

[222] J. L. Peterson. *Petri Net Theory and the Modeling of Systems*. Prentice Hall, 1 edition, 1981.

[223] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, University of Bonn, 1962. (In German).

[224] I. Phillips and I. Ulidowski. Reversibility and models for concurrency. In *Essays on Algebraic Process Calculi*, volume 192(1) of *ENTCS*, pages 93–108, 2007.

[225] I. Phillips and I. Ulidowski. Reversing algebraic process calculi. *J. Log. Algebr. Program.*, 73(1-2):70–96, 2007.

[226] I. Phillips and I. Ulidowski. Reverse bisimulations on stable configuration structures. In *Proceedings of Structural Operational Semantics (SOS 2009)*, volume 18, pages 62–76. EPTCS, 2010.

[227] I. Phillips and I. Ulidowski. A logic with reverse modalities for history-preserving bisimulations. In *EXPRESS*, volume 64 of *EPTCS*, pages 104–118, 2011.

[228] I. Phillips and I. Ulidowski. A hierarchy of reverse bisimulations on stable configuration structures. *Mathematical Structures in Computer Science*, 22(2):333–372, 2012.

[229] I. Phillips and I. Ulidowski. Reversibility and asymmetric conflict in event structures. In *CONCUR*, volume 8052 of *LNCS*, pages 303–318. Springer, 2013.

[230] I. Phillips and I. Ulidowski. An event identifier logic. *Mathematical Structures in Computer Science*, 24:1–51, 2014.

[231] I. Phillips and I. Ulidowski. Reversibility and asymmetric conflict in event structures. *Journal of Logical and Algebraic Methods in Programming*, 84(6):781–805, 2015.

[232] I. Phillips, I. Ulidowski, and S. Yuen. A reversible process calculus and the modelling of the ERK signalling pathway. In *RC*, volume 7581 of *LNCS*, pages 218–232. Springer, 2012.

[233] I. Phillips, I. Ulidowski, and S. Yuen. Modelling of bonding with processes and events. In *Proceedings of the 5th International Conference on Reversible Computation, RC 2013*, volume 7948 of *LNCS*, pages 141–154. Springer-Verlag, 2013.

[234] I. Phillips, I. Ulidowski, and S. Yuen. Modelling of bonding with processes and events. In *RC*, volume 7948 of *LNCS*, pages 141–154. Springer, 2013.

[235] I. C. C. Phillips and I. Ulidowski. Reversing algebraic process calculi. In *FoSSaCS*, volume 3921 of *LNCS*, pages 246–260. Springer, 2006.

[236] J.-E. Pin. On reversible automata. In *Latin 1992: Theoretical Informatics*, volume 583 of *LNCS*, pages 401–416. Springer, 1992.

[237] I. V. Puthoor. *Theory and Applications of Quantum Process Calculus*. PhD thesis, University of Glasgow, 2015.

[238] L. Recalde, E. Teruel, and M. Silva. Modeling and analysis of sequential processes that cooperate through buffers. *IEEE Transactions on Robotics and Automation*, 14(2):267–277, 1998.

[239] W. Reisig. *Petri Nets*, volume 4 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1982.

[240] S. A. Reveliotis and J. Y. Choi. Designing reversibility-enforcing supervisors of polynomial complexity for bounded Petri nets through the theory of regions. In S. Donatelli and P. S. Thiagarajan, editors, *Petri Nets and Other Models of Concurrency - ICATPN 2006, 27th International Conference on Applications and Theory of Petri Nets and Other Models of Concurrency, Turku, Finland, June 26-30, 2006, Proceedings*, volume 4024 of *Lecture Notes in Computer Science*, pages 322–341. Springer, 2006.

[241] S. A. Romanenko. A Compiler Generator Produced by a Self-Applicable Specializer Can Have a Surprisingly Natural and Understandable Structure. In D. Bjørner, A. P. Ershov, and N. D. Jones, editors, *Proceedings of the International Workshop on Partial Evaluation and Mixed Computation*, pages 445–463. North-Holland, Amsterdam, 1988.

[242] S. A. Romanenko. Inversion and metacomputation. In *Partial Evaluation and Semantics-Based Program Manipulation*, pages 12–22. Sigplan Notices, 26(9), ACM, New York, September 1991.

[243] K. Sabnani and A. Dahbura. A protocol test generation procedure. *Comput. Netw. ISDN Syst.*, 15(4):285–297, Sept. 1988.

[244] V. Salo. Groups and monoids of cellular automata. In J. Kari, editor, *Cellular Automata and Discrete Complex Systems*, volume 9099 of *Lecture Notes in Computer Science*, pages 17–45. Springer Berlin Heidelberg, 2015.

[245] V. Salo and I. Törmä. Computational aspects of cellular automata on countable sofic shifts. *Mathematical Foundations of Computer Science 2012*, pages 777–788, 2012.

[246] J. P. Secher and M. H. Sørensen. From checking to inference via driving and dag grammars. In *PEPM*, pages 41–51. ACM, 2002. Also published in SIGPLAN Notices (vol. 37).

[247] P. Selinger. Dagger compact closed categories and completely positive maps. *Electronic Notes in Theoretical Computer Science*, 170:139–163, 2007.

[248] P. Selinger. A survey of graphical languages for monoidal categories. In B. Coecke, editor, *New Structures for Physics*, volume 813 of *Lecture Notes in Physics*, pages 289–355. Springer, 2011.

[249] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 1948.

[250] A. Simão and A. Petrenko. Fault coverage-driven incremental test generation. *The Computer Journal*, 53:1508—1522, 2010.

[251] K. Sutner. De Bruijn graphs and linear cellular automata. *Complex Systems*, 5(1):19–30, 1991.

[252] L. Szilard. Über die Entropieverminderung in einem thermodynamischen System bei Eingriffen intelligenter Wesen. *Zeitschrift für Physik A Hadrons and Nuclei*, 53:840–856, 1929.

[253] E. Teruel, J. M. Colom, and M. Silva. Choice-free Petri nets: a model for deterministic concurrent systems with bulk services and arrivals. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, 27(1):73–83, 1997.

[254] E. Teruel and M. Silva. Liveness and home states in equal conflict systems. In M. A. Marsan, editor, *Application and Theory of Petri Nets 1993*, volume 691 of *Lecture Notes in Computer Science*, pages 415–432. Springer-Verlag, 1993.

[255] F. Tiezzi and N. Yoshida. Towards reversible sessions. In *PLACES*, volume 155 of *EPTCS*, pages 17–24, 2014.

[256] F. Tiezzi and N. Yoshida. Reversible session-based pi-calculus. *J. Log. Algebr. Meth. Program.*, 84(5):684–707, 2015.

[257] F. Tiezzi and N. Yoshida. Reversing single sessions. In *RC*, volume 9720 of *LNCS*, pages 52–69. Springer, 2016.

[258] T. Toffoli. Computation and construction universality of reversible cellular automata. *Journal of Computer and System Sciences*, 15:213–231, 1977.

[259] T. Toffoli and N. Margolus. *Cellular Automata Machines: A New Environment for Modeling*. MIT press, 1987.

[260] I. Ulidowski, I. Phillips, and S. Yuen. Concurrency and reversibility. In *RC*, volume 8507 of *LNCS*, pages 1–14. Springer, 2014.

[261] W. Vogler. Live and bounded free choice nets have home states. *Petri Net Newsletter*, 32:18–21, 1989.

[262] J. von Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, 1966.

[263] S. Wacker and T. Worsch. On completeness and decidability of phase space invertible asynchronous cellular automata. *Fundamenta Informaticae*, 126(2-3):157–181, 2013.

[264] P. Wang, Z. Ding, and H. Chai. An algorithm for generating home states of Petri nets. *Journal of Computational Information Systems*, 7(12):4225–4232, 2011.

[265] J. Wiedermann. Weak parallel machines: A new class of physically feasible parallel machine models. In *Mathematical Foundations of Computer Science 1992, 17th International Symposium, MFCS 1992, Proceedings*, pages 95–111, 1992.

[266] G. Winskel. Event structures. In *Petri Nets: Central Models and Their Properties*, volume 255 of *Lecture Notes in Computer Science*, pages 325–392. Springer, 1986.

[267] K. Yasuda, T. Kubota, and Y. Kakutani. Observational equivalence using schedulers for quantum processes. In B. Coecke, I. Hasuo, and P. Panangaden, editors, *Proceedings of the 11th workshop on Quantum Physics and Logic, QPL 2014, Kyoto, Japan, 4-6th June 2014.*, volume 172 of *EPTCS*, pages 191–203, 2014.

[268] M. Ying. Floyd-hoare logic for quantum programs. *ACM Trans. Program. Lang. Syst.*, 33(6):19, 2011.

[269] M. Ying and Y. Feng. An algebraic language for distributed quantum computing. *IEEE Trans. Computers*, 58(6):728–743, 2009.

[270] M. Ying, Y. Feng, R. Duan, and Z. Ji. An algebra of quantum processes. *ACM Trans. Comput. Log.*, 10(3), 2009.

[271] M. Ying, Y. Li, N. Yu, and Y. Feng. Model-checking linear-time properties of quantum systems. *ACM Trans. Comput. Log.*, 15(3):22:1–22:31, 2014.

[272] M. Ying, N. Yu, Y. Feng, and R. Duan. Verification of quantum programs. *CoRR*, abs/1106.4063, 2011.

[273] M. Ying, N. Yu, Y. Feng, and R. Duan. Verification of quantum programs. *Sci. Comput. Program.*, 78(9):1679–1700, 2013.

[274] S. Ying and M. Ying. Reachability analysis of quantum markov decision processes. *CoRR*, abs/1406.6146, 2014.

[275] N. Yu and M. Ying. Reachability and termination analysis of concurrent quantum programs. *CoRR*, abs/1206.1935, 2012.

[276] N. Yu and M. Ying. Reachability and termination analysis of concurrent quantum programs. In M. Koutny and I. Ulidowski, editors, *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 69–83. Springer, 2012.