A Framework for Heterotic Computing

Susan Stepney
Department of Computer Science, University of York, UK
susan@cs.york.ac.uk

Viv Kendon
School of Physics and Astronomy, University of Leeds, UK
V.Kendon@leeds.ac.uk

Peter Hines
Department of Computer Science, University of York, UK
phines@cs.york.ac.uk

Angelika Sebald
Department of Chemistry, University of York, UK
angelika.sebald@york.ac.uk

Computational devices combining two or more different parts, one controlling the operation of the other, for example, derive their power from the interaction, in addition to the capabilities of the parts. Non-classical computation has tended to consider only single computational models: neural, analog, quantum, chemical, biological, neglecting to account for the contribution from the experimental controls. In this position paper, we propose a framework suitable for analysing combined computational models, from abstract theory to practical programming tools. Focusing on the simplest example of one system controlled by another through a sequence of operations in which only one system is active at a time, the output from one system becomes the input to the other for the next step, and *vice versa*. We outline the categorical machinery required for handling diverse computational systems in such combinations, with their interactions explicitly accounted for. Drawing on prior work in refinement and retrenchment, we suggest an appropriate framework for developing programming tools from the categorical framework. We place this work in the context of two contrasting concepts of "efficiency": theoretical comparisons to determine the relative computational power do not always reflect the practical comparison of real resources for a finite-sized computational task, especially when the inputs include (approximations of) real numbers. Finally we outline the limitations of our simple model, and identify some of the extensions that will be required to treat more complex interacting computational systems.

1 Introduction

Classical computation theory is epitomised by the Turing machine paradigm. We are concerned with more diverse models of computation, in particular determined by the physical properties of the system used as a computer [38]. A broad range of experiments and theory is being developed to investigate the computational capabilities of chemical [26, 32, 39], biological [2, 3], quantum [36], optical [42, 40], and various analog [28, 35, 31] computational substrates. Given that we have different types of computational devices, not necessarily Turing universal, it is natural to ask how to *compose* them, and to ask about the computational power of the composition. We term such composed systems *heterotic computers*[1].

The computational power of a given physical system is determined not only by the operations available to manipulate the system, but also by the type of data that can be encoded in the system and the measurements available to decode the result of the computation. When composing different systems, information must pass between them, making these data types and measurements relevant throughout the computation. This is in contrast to classical complexity analysis, which focuses on the operations that

[1] *Heterotic*, from the Greek *heterosis*, a term in genetics meaning "hybrid vigour".

Bart Jacobs, Peter Selinger, and Bas Spitters (Eds.):
8th International Workshop on Quantum Physics and Logic (QPL 2011)
EPTCS 95, 2012, pp. 263–273, doi:10.4204/EPTCS.95.18

perform the computation, and assumes that data input and output are trivial in comparison. More care is generally taken when using non-standard computational models. For example, in quantum computing, DiVincenzo's checklist [19] first identifies a physical system that can represent a qubit, then identifies a set of operations sufficiently rich to provide universal quantum computation. Output from quantum systems is also non-trivial, since measurements cannot determine the full quantum state with certainty. Extra procedures in the algorithm are required to ensure the measurement gives a useful output with high probability. However, this analysis still focuses on the quantum processor without giving explicit account of the role of the classical control systems.

Thus, we need a framework that not only allows different models of computation to be compared and contrasted, but also allows us to compose different models and determine the resulting computational power, as motivated in [25]. In this position paper, we provide more details of the categorical tools required to accomplish this. Together with a refinement/retrenchment approach to support program development, these would provide the tools to determine the combined computational power of the heterotic computer. The paper is organized as follows: In §2 we summarise prior work on several heterotic systems: measurement-based quantum computing; NMR classical computing; qubus quantum computing. In §3 we outline the categorical framework, in the context of a simple two-layer computational architecture, and outline a semantic basis and refinement approach. In §4 we describe how to create the programming tools from this framework, using a modified refinement based method. In §5 we summarise and outline the next steps for this work.

2 Heterotic computational systems

The role of the classical controlling system in quantum computation was first noted by Josza [24], while demonstrating the equivalence of measurement-based and teleportation-based quantum computing schemes. In measurement-based quantum computing (MBQC), also known as cluster state, and as one-way, quantum computing [33], an entangled resource of many qubits is prepared, then the computation proceeds by measuring the qubits in turn. The outcomes from the measurements feed forward to determine the type of measurement performed on the next qubits (figure 1a). It was not until 2009 that Anders and Browne [4] realised that the classical computation required to control and feed forward information in MBQC is a crucial part of the computational power. Applying measurements without feed-forward is efficiently classically simulable, as is (trivially) the classical part of the computation. However, the combination of the two is equivalent to the quantum circuit model, which is not (efficiently) classically simulable. Thus the combination of two or more systems, to form a new computational system composed of several layers, can be in a more powerful computational class than the layers acting separately.

Equivalent examples have been described in the realm of classical unconventional computation. In experiments using liquid state NMR to perform simple gate logic [34] such as NAND, the instruments controlling the NMR pass the outputs of one gate through to the inputs of the next (figure 1b). As with MBQC, these controls play an essential role in the computation, but by themselves do not perform the gate logic. Using NMR to do classical computing involves choosing a subset of the available parameters suitable for representing classical bits, and restricting the operations to keep the spin ensemble in a fully determined classical states. In this way, more robust operations are obtained at the expense of not exploiting the full capabilities of the physical system. Prior work on computation using NMR mostly deals with implementations of quantum computations, predominantly based on solution-state NMR experiments [23], with some examples exploiting solid-state NMR [17]. As a step towards characterizing the computational power of the NMR system, Bechmann et al [10] have produced a preliminary clas-

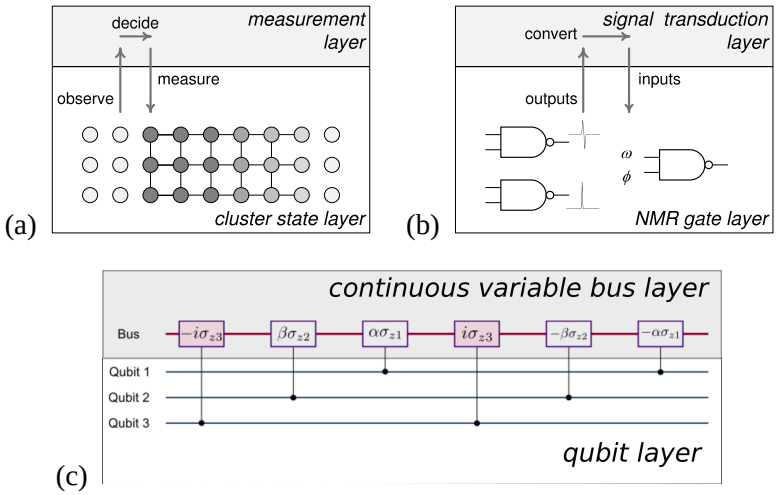

Figure 1: (a) Measurement-based quantum computer. The base layer is a cluster state. The control layer performs measurements on the base layer, thereby changing its state; the control layer uses the observed results of a measurement to decide what measurement to perform next. (b) Classical NMR computer [34]. The base layer gates are implemented as NMR experiments: inputs are frequencies ω and phase delays ϕ; outputs are the integrated output signal. The control layer performs "signal transduction": taking the integrated output, interpreting it as a 0 or 1, and converting that to the appropriate physical input signal. (c) Qubus quantum computer. The base layer is qubits, the control layer is a coherent state, which can interact with several qubits at the same time, enacting the gates between the qubits. There are no measurements in this fully quantum example, the qubit state determines the interaction with the bus, which in turn changes the qubit state according to the externally chosen order in which it interacts with the qubits.

sification of the experimental NMR parameters for implementing classical logic gates. This work has been extended to take advantage of the inherently continuous nature of the NMR parameter space of non-coupled spin species [11] by implementing continuous gates, so the combined system performs an analog computation. However, the extent to which the control layer contributes to the computational power of quantum or classical NMR computing has yet to be analysed.

The theory of ancilla-based quantum computation [5] has been abstracted and developed from MBQC, into a framework where a quantum system (ancilla) controls another quantum system (the qubits), with or without measurement of the ancilla system during the computation. This framework is capable of modelling many types of hybrid quantum computing architectures. When the role of the ancilla system is played by a continuous variable quantum system instead of a qubit or qudit (d-dimensional quantum system) further efficiencies become available. The qubus quantum computer uses a coherent state as the bus, which has two quadratures, which act as two coupled continuous variable quantum systems. This type of ancilla can interact with many qubits at the same time, allowing savings in the number of basic operations required for gate operations [14] and for building cluster states [22, 13]. Figure 1c shows a sequence of six operations that performs four gates, one between each possible pair of the three qubits. Each gate performed separately would require two operations, thus this sequence saves at least two operations over standard methods, more if the qubits have to be swapped to adjacent positions for direct gates. Typically, this provides polynomial reductions in the number of elementary operations required for a computation, when compared with interacting the qubits directly.

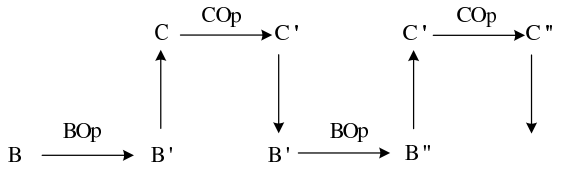

Figure 2: The stepwise interactions between a base computation (state B, state change BOp) and a controller computation (state C, state change COp): the input to one is the output from the other.

3 Towards a categorical heterotic framework

The examples in §2 can all been depicted with the same structure of a base layer and a control layer (figure 2). This can be generalised to multiple layers, each controlling the layer below, and being controlled by the one above, and to layers with feedback loops that couple non-adjacent layers. Here we focus on the simplest heterotic model, in which just two computers are coupled, one controlling the other. For now, we take as given the particular division into layers: we do not need this to be a unique decomposition in what follows.

The pattern of computation and communication alternates between the two layers (figure 2). In this basic model, the state of one layer does not change during the computation by the other (for example, the control layer remains in state C' as the base layer evolves from B' to B''). The basic model allows a physical implementation where the state continues to evolve, if its subsequent computation depends only on its input (either it is essentially "reset" to the previous state, or the input fully determines what happens next). This case holds for our motivating examples in figures 1a and 1b, although they have not yet been explicitly cast in the framework. In the qubus example, figure 1c, the layers shown evolve only while interacting with each other. However, single qubit gates applied directly to the qubits can be inserted whenever the qubus is not connected to the qubit in question, which would then be an example of two separate controlling layers doing different tasks. Furthermore, the coherent state (a quantum state) acting as the bus itself has a classical control layer (not depicted), which determines the parameters (α, β) in the interactions with the qubits. This architecture thus goes beyond our simple starting point of two coupled computers, and serves to remind us that extensions to the basic model will be required.

One of the goals is a form of *refinement calculus for heterotic computers*, suitable for use by the working programmer, to enable the full power of such systems to be exploited. However, producing such a framework first requires theoretical input. In particular, we need a suitable form of semantics on which the refinement calculus is based. Such models exist for individual systems, for example, classical analog computation has been modelled in several ways, from the traditional approaches based on differential equations, to interval-based analyses relying on domain theory. Classical probabilistic computation can be modelled via categories of stochastic relations, and non-determinism frequently requires categories of relations, or constructions based on the power set functor. For heterotic computing, the theoretical challenge is to give a formal description of how such systems may interact in non-trivial ways. Due to the wide range of heterotic computing systems under consideration, we aim for an abstract categorical semantics, and seek concrete instantiations where appropriate.

Given two dissimilar systems A and B, and models of each of these in distinct categories $\mathscr{C}_A$ and $\mathscr{C}_B$, we require a formal setting in which both the joint system, and the non-trivial interactions between systems A and B, may be modelled. If we wish to model a joint system *without* considering interactions, the product category $\mathscr{C}_A \times \mathscr{C}_B$ is the natural choice; however, for our purposes, it is entirely inappropriate. The real object of study (and, we claim, source of computational power) is found in the non-trivial

interactions between the subsystems.

How, then should we describe interactions between computing devices whose models are to be found in distinct categories? One approach would be to find some larger encompassing category, sufficiently broad and general to model both devices (similar to the way that both classical probabilities and complex phases may be combined in the density matrix formalism of quantum mechanics). However, the downside of this approach is that, with highly dissimilar devices, the required framework must be excessively abstract or general. There is also the more philosophical objection that this approach would be trying to treat our interacting systems as a single system in some more general setting, missing the motivation of studying the *interaction* of distinct systems for its source of computing power.

So instead, to model interactions between systems A and B, we rely on some structure-preserving map from models of system A to models of system B, and vice versa. These must be functors $\Gamma : \mathscr{C}_A \to \mathscr{C}_B$ and $\Delta : \mathscr{C}_B \to \mathscr{C}_A$. The question is, what further categorical properties must these be expected to display?

As a motivating example, we consider categorical structures that are at the core of many computing systems, and consider how they can be either generalised or relaxed, in order to deal with systems based on interacting distinct systems. In categorical models of logic and computation, the notion of a closed category – usually monoidal closed – is often fundamental. In logical systems, monoidal closure provides the structure necessary to model cut-elimination, and given a computational interpretation of logical systems (commonly via the Curry-Howard isomorphism) this interprets as β-reduction in lambda calculus [27]. Other logical or computational interpretations are available, from compositionality in models of Turing machines [21], to the essential categorical structure of teleportation in quantum computation [1].

A monoidal category $\mathscr{C}$, has a functor (the *monoidal tensor*) $\otimes : \mathscr{C} \times \mathscr{C} \to \mathscr{C}$, satisfying $(A \otimes B) \otimes C \cong A \otimes (B \otimes C)$ together with a unit object I satisfying $A \otimes I \cong A \cong I \otimes A$. (The families of arrows exhibiting these isomorphisms must satisfy additional *coherence* and *naturality* conditions; see [29] for more details). A monoidal category is *monoidal closed* when there also exists a functor (the *internal hom*) $[\_ \to \_] : \mathscr{C}^{op} \times \mathscr{C} \to \mathscr{C}$ that satisfies

$$\mathscr{C}(A \otimes B, C) \;\cong\; \mathscr{C}(B, [A \to C]) \tag{1}$$

This is a canonical example of an adjunction. Further, in the very special case where the system is *untyped* (so all objects of $\mathscr{C}$, excluding the unit object, are isomorphic), we recover the familiar untyped equations $D \;\cong\; D \otimes D \;\cong\; [D \to D]$ providing models of *universal computation* (e.g. the C-monoids of [27] or the untyped compact closure of [20]).

For our purposes, monoidal closure, in either its typed or untyped form, is too strong: it describes situations where the computation is carried out in a single homogeneous system. Further, we do not expect, or require, universal computation from our heterotic systems. Instead, we take the notion of an adjunction between two functors as primitive, and expect to recover more familiar models of computation in the special case where the interacting systems are identical.

The notion of an adjunction is simply a categorification of the concept of a Galois connection, thus two functors $\Gamma : \mathscr{C}_A \to \mathscr{C}_B$ and $\Delta : \mathscr{C}_B \to \mathscr{C}_A$ form an adjoint pair when $\mathscr{C}_A(\Gamma(X), Y) \;\cong\; \mathscr{C}_B(X, \Delta(Y))$, for all $X \in Ob(\mathscr{C}_A)$, $Y \in Ob(\mathscr{C}_B)$. The duality provided by such an adjunction allows us to model the mutual update of system A by system B and system B by system A, without requiring that system B is fully able to simulate the behaviour of system A, or vice versa. We are thus able to capture the sometimes hidden symmetries we expect to find within such interactions.

For concrete examples, we expect much more categorical structure; we are not claiming that the theory of adjunctions in itself will provide enough structure to give categorical semantics of heterotic systems. However, we take the existence of a suitable adjunction, between categories modelling dissimilar systems, as the basic defining characteristic of a heterotic system. Each concrete example will depend

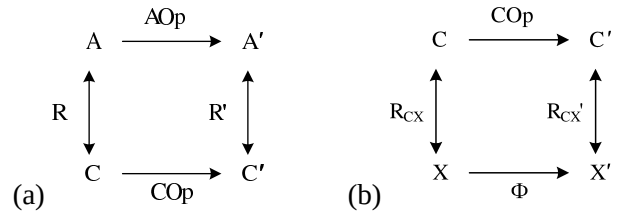

Figure 3: (a) A simulation, used to prove refinement; (b) Physical and computational layer relationship

on the specific details of the two interacting systems. An illustrative example is available in the categorical semantics approach of Abramsky and Coecke [1], where an adjunction (via its characterisation as unit/co-unit maps in a 2-category setting) is used to describe creation of quantum systems from classical data, and measurement of quantum systems (resulting in classical information).

Thus, it appears that relatively simple category theory provides ready-made abstract conditions suitable for describing the mutual update of distinct systems in heterotic computing, along with real concrete examples of how this works in certain settings.

4 A heterotic refinement framework

Given some suitable semantic framework, such as the one outlined above, it is necessary to cast it in a form suitable for enabling the working programmer to analyse and develop novel heterotic systems in (relatively) familiar ways. We suggest that a classical refinement framework is more appropriate than, say, a process algebra approach, since this is more accessible and familiar to the working programmer.

State-and-operation refinement is the classical computational approach to program development. It takes an abstract, possibly non-deterministic, specification of a state A evolving under a sequence of operations AOp, and *refines* it (reducing non-determinism, changing data types) into a more concrete implementation with state C and operations COp, with the abstract state A *retrieved* from the concrete state C through the retrieve relation R (figure 3a). We have the refinement correctness requirement (ignoring non-determinism here for simplicity) that the diagram *commute* (we get the same value for C' either way round):

$$R'(AOp(A)) = COp(R(A)) \tag{2}$$

Usually the process of refinement stops at a computational level suitably concrete to allow implementation, such as a mid-level programming language. It can in principle be carried further. Here we need to consider it all the way down to the physical implementation, since we are interested in non-classical execution models. So we continue refining from C down to the physical level, with a state X, that evolves under the laws of physics, Φ. The physical state variables in X are again retrieved through relation R_{CX} as computational state variables in C (figure 3b). Refinement reduces non-determinism until we reach a completely deterministic implementation. We classically think of the resolution of non-determinism being under the control of the programmer, but when we reach the physical layer we may be left with intrinsic non-determinism. In the case of quantum computation, while unitary quantum evolution is deterministic, measurement of quantum systems in general is not. The programmer can either arrange for the algorithm to present a final state with a deterministic measurement outcome (derandomization), or, accept that the computation may need to be repeated to increase the probability of obtaining the required outcome. Note that the induced computation COp depends on both the physical system Φ and the viewing interpretation R_{CX}. We would like this diagram to commute (to get the same value for X' either way

round), but there will be errors (measurement, noise)[2]. So we can at best require the inexact commutation

$$R_{CX}{}'(COp(C)) = \Phi(R_{CX}(C)) \pm \varepsilon \tag{3}$$

Retrenchment [6, 7, 8, 9] is a form of inexact refinement. It allows deviations from exact refinements by use of various forms of *concedes* clauses; analysis of the retrenchment concessions provides insight into the way an implementation deviates from a pure refinement. In particular, retrenchment has been applied to developing discrete implementations of real number specifications [8], and to finite implementations of unbounded natural number specifications, which are necessarily inexact. Also, it has been suggested as a laboratory for analysing and understanding emergent behaviour of complex systems [6].

Retrenchment has its critics in the purist refinement community, but we have argued elsewhere [7] that these criticisms are invalid in the context of real world engineering developments, even in the classical computing model. Here we claim that (some suitably posed form of) retrenchment is appropriate for casting non-exact computations in unconventional substrates in a refinement-like framework. It would be used to analyse the size, nature, and propagation of errors.

The usual classical refinement correctness rules allow inputs to and outputs from the operations, but require these to be the same at the abstract and concrete levels. In previous work [16], we have generalised these rules to allow refinement of i/o, too. This necessitated the introduction of a *finalisation* step, that can be interpreted as the definition of the observation made on the system. There is an *initialisation* step, that we have extended to interpret inputs analogously. The finalisation of the most abstract level is usually the identity (we see the "naked" abstract i/o); more concrete implementations have more sophisticated finalisations (eg, we see a bit stream, but view it, finalise it, as an integer) [15]. The correctness rule (again, ignoring non-determinism) is

$$AFin(A) = CFin(R(A)) \tag{4}$$

This work has also been extended to the retrenchment arena.

A form of i/o refinement is necessary to move between physical i/o variables and computational i/o variables. For example, in the case of the NMR adder [34]: the physical level is the NMR; the computational level is the NAND gate; the initialisation is interpreting a frequency and a phase delay as a bit; the finalisation is observing an integrated signal as a bit. For this form of initialisation/finalisation to work in the analysis, it has to be possible *in principle* to provide all the inputs at the start of the computation, and to observe (a record of) all the outputs at the end. This cannot be done for the individual layers of the heterotic computation, where the output from one layer becomes the input to the other (it is closer to a Wegner interaction machine architecture [41]) but can for the overall computation, so we need to be careful about how we set up the analysis, and precisely what we define as i/o. This step is crucial in our heterotic framework, since, as stated earlier, the encoding and decoding processes (formalised as initialisation and finalisation) are non-trivial in general.

We have an additional step in the NMR example [34], where the physical inputs and outputs are of different types, but the output from one step becomes the input to the next. We perform a *signal transduction* step here (integrals over Fourier transforms transduced to phases, that preserves the initialisation/finalisation interpretations). This does not have an analogue in the refinement scenario, because that does not include any link between the outputs of one operation and the inputs of the next. This is important in the context of heterotic computing, as there is potentially significant computation applied to outputs to produce the next inputs. This computation is performed by the other part of the computer.

[2]Classical digital hardware is extremely engineered to ensure an exact boolean implementation; this exactness cannot necessarily be assumed in the more general case.

The base and controller levels can be implemented (refined) separately. For example, in the quantum cluster state and the classical controller (figure 1a), the state is set up initially, and the only operation performed in the base layer is measurement; which measurement to perform is determined in the classical controller level based on previous measurement results. The measurement itself changes the state, which is part of the computation. In NMR (figure 1b), where the base level is the NMR gates; the controller level is mere signal transduction – this shows that there is no sharp separation between the i/o refinement and the computation (in this case it can be done in either).

These concrete models can be used as the basis for developing a suitable form of refinement calculus. Possibly the closest pre-existing work relating to this is the use of weakest precondition semantics to study Grover's algorithm developed by d'Hondt and Panagaden [18] — in particular, the way that a hybrid quantum/probabilistic setting is modelled by the density matrix formalism. This gives a specific case of the type of underlying logical rules that need to be preserved by the refinement calculus, by analogy with the way that traditional program refinement preserves the Hoare logic. However, in each concrete setting, the behaviour/logic preserved by the refinement process will be different, and the formal calculus produced in each case will be heavily dependent on the underlying categorical models. Moreover, for non-discretised systems, this relevant refinement calculus would need to be extended to a retrenchment approach to allow a well-defined and principled form of inexact refinement. This would include analysis of propagation of errors [12] (due to noise, and to drift), and techniques for correction and control of these errors.

5 Discussion and conclusions

We have described a novel computational framework, heterotic computation, that can be used to combine computational systems from different implementation paradigms in a principled and controlled manner, to produce a computational system qualitatively different from either in isolation. We have outlined a semantic and refinement framework that could be used to support such an approach.

One goal of such a framework is to analysise the efficiency of a computational system. Here we take a broad view of "efficiency": it covers both the traditional scaling and complexity classes, and also covers issues of real-time performance on real world scale problems. Both views are important, and they do not necessarily coincide, especially in combinations of disparate physical systems each being exploited for its own particular computational capabilities. As an example, the quantum community is developing "hybrid computing" [37, 36, 30], to create practical quantum systems that can compute something non-trivial before errors come to dominate. There efficiency gains from the theoretical complexity point of view are considered later, only once the abstract theory is tackled (for example, MBQC vs ancilla-driven quantum computation). The heterotic framework, in both its categorical semantics and its refinement/retrenchment calculus, allows for a range of efficiency considerations, because it allows analysis of the computational processes and error propagation in all the relevant parts of the system: the individual layers, their interactions, and the overall system. From this, both the complexity theoretic efficiency and the practical efficiency can be derived.

This is only the first step in such heterotic computation. We have mentioned several areas that would need enhancement to the simple framework we have started with: where the base layer continues its computation whilst the controlling layer is working, and where there is more than one layer. A range of dynamical systems will contain continuously evolving layers; one of the things the controlling layer will need to decide is when to probe/perturb the base layer, to exploit its dynamics. Additionally, further forms of parallelism also need to be added to the framework.

We believe the heterotic approach is needed to ensure that the many forms of unconventional computation can be exploited fully. Each individual paradigm no longer need be distorted to achieve Turing-completeness. Instead, different components can be combined to form a more powerful system, with each component doing what it does naturally, and best.

Acknowledgments:

VK is funded by a UK Royal Society University Research Fellowship. We thank Matthias Bechmann, Rob Wagner, and Katherine Brown for useful discussions.

References

[1] S. Abramsky & B. Coecke (2004): *A categorical semantics of quantum protocols*. *Proc. IEEE Symp. Logic In Comp. Sci.* , pp. 415–425, doi:10.1109/LICS.2004.1319636.

[2] Andrew Adamatzky (2007): *Physarum machines: encapsulating reaction-diffusion to compute spanning tree*. *Naturwissenschaften* 94(12), pp. 975–980, doi:10.1007/s00114-007-0276-5.

[3] Martyn Amos (2005): *Theoretical and Experimental DNA Computation*. Springer, ISBN 978-3-642-08504-8.

[4] Janet Anders & Dan Browne (2009): *Computational power of correlations*. *Phys. Rev. Lett.* 102, p. 050502, doi:10.1103/PhysRevLett.102.050502.

[5] Janet Anders, Daniel K. L. Oi, Elham Kashefi, Dan E. Browne & Erika Andersson (2010): *Ancilla-driven universal quantum computation*. *Phys. Rev. A* 82(2), p. 020301, doi:10.1103/PhysRevA.82.020301.

[6] Richard Banach, Czeslaw Jeske, Simon Fraser, Richard Cross, Michael Poppleton, Susan Stepney & Steven King (2004): *Approaching the Formal Design and Development of Complex Systems: The Retrenchment Position*. In: *WSCS, IEEE ICECCS'04*. Available at http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.9231.

[7] Richard Banach, Czeslaw Jeske, Mike Poppleton & Susan Stepney (2007): *Retrenching the Purse*. *Fundamenta Informaticae* 77, pp. 29–69. Available at http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.61.8631.

[8] Richard Banach & Mike Poppleton (1998): *Retrenchment: an engineering variation on refinement*. In: *2nd Intl. B Conference, LNCS* 1393, Springer, pp. 129–147, doi:10.1007/BFb0053358.

[9] Richard Banach, Mike Poppleton, Czeslaw Jeske & Susan Stepney (2007): *Engineering and theoretical underpinnings of retrenchment*. *Sci. Comp. Prog.* 67(2-3), pp. 301–329, doi:10.1016/j.scico.2007.04.002.

[10] M. Bechmann, A. Sebald & S. Stepney (2011): *Boolean logic-gate design principles in unconventional computers: an NMR case study*. *International J. Unconventional Computing* In press.

[11] Matthias Bechmann, Angelika Sebald & Susan Stepney (2010): *From binary to continuous gates – and back again*. In: *ICES 2010, LNCS* 6274, Springer, pp. 335–347, doi:10.1007/978-3-642-15323-5_29.

[12] Ed Blakey (2010): *Unconventional complexity measures for unconventional computers*. *Natural Computing* doi:10.1007/s11047-010-9226-9.

[13] K. L. Brown, C. Horsman, V. M. Kendon & W. J. Munro (2011): *Layer by layer generation of cluster states*. Available at http://arxiv.org/abs/1111.1774v1.

[14] Katherine L. Brown, Suvabrata De, Viv Kendon & William J. Munro (2011): *Ancilla-based quantum simulation*. *New J. Phys.* 13, p. 095007, doi:10.1088/1367-2630/13/9/095007.

[15] John A. Clark, Susan Stepney & Howard Chivers (2005): *Breaking the Model: finalisation and a taxonomy of security attacks*. *ENTCS* 137(2), pp. 225–242, doi:10.1016/j.entcs.2005.04.033.

[16] D. Cooper, S. Stepney & J. Woodcock (2002): *Derivation of Z Refinement Proof Rules: forwards and backwards rules incorporating input/output refinement*. Technical Report YCS-2002-347, Department of Computer Science, University of York. Available at http://www.cs.york.ac.uk/ftpdir/reports/2002/YCS/347/YCS-2002-347.pdf.

[17] David G. Cory et al. (2000): *NMR Based Quantum Information Processing: Achievements and Prospects*. *Fortschritte der Physik* 48(9–11), pp. 875–907, doi:10.1002/1521-3978(200009)48:9/11.

[18] E. d'Hondt & P. Panangaden (2006): *Quantum weakest preconditions*. *Math. Struct. Comp. Sci.* 16(3), pp. 429–451, doi:10.1017/S0960129506005251.

[19] David P. DiVincenzo (2000): *The Physical Implementation of Quantum Computation*. *Fortschritte der Physik* 48(9–11), pp. 771–783, doi:10.1002/1521-3978(200009)48:9/11. ArXiv:quant-ph/0002077v3.

[20] Peter Hines (1999): *The categorical theory of self-similarity*. *Theory and Applications of Categories* 6, pp. 33–46. Available at http://emis.math.ca/journals/TAC/volumes/6/n3/n3.pdf.

[21] Peter Hines (2003): *A categorical framework for finite state machines*. *Mathematical Structures in Computer Science* 13, pp. 451–480, doi:10.1017/S0960129503003931.

[22] Clare Horsman, Katherine L. Brown, William J. Munro & Vivien M. Kendon (2011): *Reduce, reuse, recycle for robust cluster-state generation*. *Phys. Rev. A* 83(4), p. 042327, doi:10.1103/PhysRevA.83.042327.

[23] Jonathan A. Jones (2011): *Quantum computing with NMR*. *Progress in Nuclear Magnetic Resonance Spectroscopy* 59, pp. 91–120, doi:10.1016/j.pnmrs.2010.11.001.

[24] Richard Jozsa (2005): *An introduction to measurement based quantum computation*. Available at http://arxiv.org/abs/quant-ph/0508124.

[25] V. Kendon, A. Sebald, S. Stepney, Matthias Bechmann, Peter Hines & Robert C. Wagner (2011): *Heterotic computing*. In: *Unconventional Computation, LNCS*, 6714, Springer, pp. 113–124, doi:10.1007/978-3-642-21341-0_16.

[26] L. Kuhnert, K. Agladze & V. Krinsky (1989): *Image processing using light-sensitive chemical waves*. *Nature* 337, pp. 244–247, doi:10.1038/337244a0.

[27] J. Lambek & P. J. Scott (1988): *An introduction to higher-order categorical logic*. *Cambridge Studies in Advanced Mathematics* 7, Cambridge University Press, ISBN 9780521356534.

[28] Seth Lloyd & Samuel L Braunstein (1999): *Quantum computation over continuous variables*. *Phys. Rev. Lett.* 82, p. 1784, doi:10.1103/PhysRevLett.82.1784.

[29] Saunders Mac Lane (1971): *Categories for the working mathematician*. Graduate Texts in Mathematics, 1st Ed., Springer Verlag, ISBN 0387900357.

[30] G. J. Milburn, S. Schneider & D. F. V. James (2000): *Ion Trap Quantum Computing with Warm Ions*. *Fortschr. Phys.* 48, pp. 801–810, doi:10.1002/1521-3978(200009)48:9/11<801::AID-PROP801>3.0.CO;2-1.

[31] Jonathan W. Mills (2008): *The nature of the Extended Analog Computer*. *Physica D: Nonlinear Phenomena* 237(9), pp. 1235–1256, doi:10.1016/j.physd.2008.03.041.

[32] Ikuko N. Motoike & Andrew Adamatzky (2005): *Three-valued logic gates in reaction-diffusion excitable media*. *Chaos, Solitons & Fractals* 24(1), pp. 107–114, doi:10.1016/S0960-0779(04)00461-8.

[33] Robert Raussendorf & Hans J Briegel (2001): *A One-Way Quantum Computer*. *Phys. Rev. Lett.* 86, pp. 5188–5191, doi:10.1103/PhysRevLett.86.5188.

[34] M. Roselló-Merino, M. Bechmann, A. Sebald & S. Stepney (2010): *Classical computing in nuclear magnetic resonance*. *International J. Unconventional Computing* 6(3–4), pp. 163–195. Available at http://www-users.cs.york.ac.uk/susan/bib/ss/nonstd/ijnmc09.pdf.

[35] D. Silva Graça (2004): *Some Recent Developments on Shannon's GPAC*. *Math. Log. Quart.* 50(4–5), pp. 473–485, doi:10.1002/malq.200310113.

[36] T. P. Spiller, W. J. Munro, S. D. Barrett & P. Kok (2005): *An introduction to quantum information processing: applications and realisations*. *Comptemporary Physics* 46, p. 407, doi:10.1080/00107510500293261.

[37] T. P. Spiller, Kae Nemoto, Samuel L. Braunstein, W. J. Munro, P. van Loock & G. J. Milburn (2006): *Quantum Computation by Communication*. *New J. Phys.* 8, p. 30, doi:10.1088/1367-2630/8/2/030.

[38] Susan Stepney (2008): *The Neglected Pillar of Material Computation*. *Physica D: Nonlinear Phenomena* 237(9), pp. 1157–1164, doi:10.1016/j.physd.2008.01.028.

[39] Ágota Tóth & Kenneth Showalter (1995): *Logic gates in excitable media*. *J. Chem. Phys* 103, pp. 2058–2066, doi:10.1063/1.469732.

[40] Rodney S. Tucker (2010): *The role of optics in computing*. *Nature Photonics* 4, p. 405, doi:10.1038/nphoton.2010.162.

[41] Peter Wegner (1997): *Why interaction is more powerful than algorithms*. *CACM* 40, pp. 80–91, doi:10.1145/253769.253801.

[42] Damien Woods & Thomas J. Naughton (2008): *Parallel and Sequential Optical Computing*. In: *Optical SuperComputing, LNCS* 5172, Springer, pp. 70–86, doi:10.1007/978-3-540-85673-3_6.