Heterotic Computing

Viv Kendon[1], Angelika Sebald[2], Susan Stepney[3],
Matthias Bechmann[2], Peter Hines[3], and Robert C. Wagner[1]

[1] School of Physics and Astronomy, University of Leeds, UK
[2] Department of Chemistry, University of York, UK
[3] Department of Computer Science, University of York, UK

Abstract. Non-classical computation has tended to consider only single computational models: neural, analog, quantum, etc. However, combined computational models can both have more computational power, and more natural programming approaches, than such 'pure' models alone. Here we outline a proposed new approach, which we term *heterotic computing*[4]. We discuss how this might be incorporated in an accessible refinement-based *computational framework* for combining diverse computational models, and describe a range of physical exemplars (combinations of classical discrete, quantum discrete, classical analog, and quantum analog) that could be used to demonstrate the capability.

1 Introduction

Classical computation 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 [27].

Given that we have different basic types of computers, not necessarily Turing universal, it is natural to ask how to *compose* them into hybrid – "heterotic" – computers, and to ask about the computational power of the composition. 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.

The structure of the paper is as follows. §2 describes two existing heterotic computers that are used to motivate our approach. §3 outlines the heterotic framework, introducing the proposed computational architecture, and outlining a semantic bases and refinement approach. §4 describes how to progress the classical nuclear magnetic resonance (NMR) computer within this framework. §5 describes an approach that challenges the proposed framework. §6 concludes.

2 Motivating examples

The computational power of a given physical system is determined by:

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

1. How much of what type of data can be encoded in the system. Eg: identical objects that can be placed in different collections can be used to encode natural numbers in a unary representation, and to do simple arithmetic by concatenation; they cannot encode negative numbers, as there is no sign bit, or zero if only non-empty collections can exist. (See also work by Blakey [8].)
2. What operations are available to manipulate the system. Eg: quantum computing, following DiVincenzo's checklist [15], first identifies a physical system that can represent a qubit, then identifies a set of operations sufficiently rich to provide universal quantum computation. A CNOT gate combined with single qubit rotations through $\pi/8$ is universal, while the gates in the Clifford group (Pauli operators plus Hadamard and CNOT) are not universal.
3. What operations are available to decode the end result. Eg: in quantum computing, the measurements available to obtain the result can reveal only a limited amount of information, making the final decoding step highly non-trivial. Additionally, they are not just observations, but can form an essential part of the computation.

Classically, only point 2 is analysed in depth, but in wider models, points 1 and 3 are also crucial. In our heterotic approach, interaction between the computational layers requires consideration of 1 and 3 within the framework.

Two motivating examples that demonstrate a layered approach to heterotic computation are described here. The first motivating example from quantum computation is the cluster state, or one-way, quantum computer. The second example is classical computation in NMR [24], where the instruments controlling the NMR form part (but not all) of the computation. We discuss these further here, to illustrate the underlying heterotic principles in action.

2.1 Cluster state computing

Anders and Browne [2] noticed that the classical computation required to control and feed forward information in a quantum cluster state computer is a crucial part of the computational power. Cluster state measurement 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. Hence the combination of layers is in a more powerful computational class than either layer alone. The base and control layer are shown in figure 1a.

2.2 Classical NMR computing

NMR experiments provide both a classical and a quantum computing layer, intimately intermixed with discrete and continuous parameters at several levels. The classical layer typically comprises the spectrometer, radio-frequency (r.f.) pulses, the detector, and the macroscopic (nuclear) magnetisation of a sample in the magnetic field, as described by density matrix formalism. The quantum layer is provided by the (individual) nuclear spin systems, with their discrete

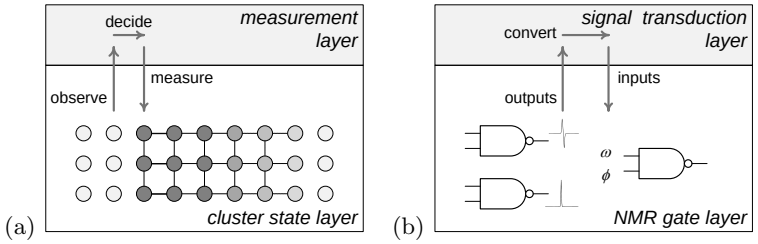

Fig. 1. (a) Cluster state computer. The base layer is a cluster state. The control layer performs measurements on the base layer, thereby changing its state; the control layer uses the observed results to decide what measurement to perform next. (b) Classical NMR computer [24]. The base layer gates are 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.

states being very well described by quantum mechanics. NMR experiments thus involve both discrete (spin states) and continuous (e.g. r.f. pulses) parameters. See figure 1b.

Most importantly, NMR experiments give us an element of choice: we can opt for a purely classical behaviour by using only samples made up from isolated (non-coupled) nuclear spins, the dynamics of which are fully described by a classical vector model (for background, see [24]), while choosing a system with coupled nuclear spins provides a mixed classical/quantum combination. Moreover, we can easily switch between the two regimes within a single experiment (e.g. de- and recoupling [16]).

Prior work on computation using NMR mostly deals with implementations of *quantum* computations, predominantly based on solution-state NMR experiments [19], with some examples exploiting solid-state NMR [13]. Prior work by ourselves has proved the suitability of NMR for classical (binary) logic gates [24], based on a range of different solution-state NMR experiments. We implemented a half-adder constructed from binary NAND gates in an NMR system with a classical control performing 'signal transduction', transforming the physical outputs of one gate to the physical inputs of the next [24]. Further, we have produced a preliminary classification of the experimental NMR parameters for implementing classical logic gates. More recently, we have extended our work to take advantage of the inherently continuous nature of the NMR parameter space of non-coupled spin species [7] by implementing continuous gates, so the combined system performs an analog computation.

The classical and quantum computational layers have thus both been demonstrated experimentally in NMR, and the rich parameter space (together with the very well developed theory of NMR and superb degrees of experimental control) provide the source of the computational power of NMR implementations. However, the extent to which the control layer contributes to the computational power of quantum or classical NMR computing has yet to be analysed.

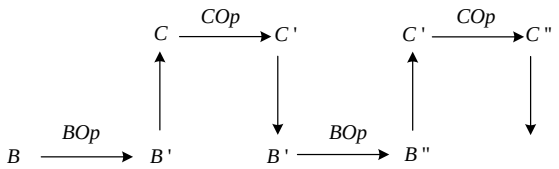

Fig. 2. The stepwise interactions between a base computation (state B, state change BOp) and a controller computation (state C, state change COp): the input to one is the output from the other.

3 Heterotic framework

3.1 Computational framework

Our basic heterotic model is two coupled computers, one 'guiding' or controlling the other. So the base machine does a step, the guiding machine looks at its output, and tells it what to do next, and so on. The pattern of computation and communication alternates between the two (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''). In a physical implementation this state might continue to evolve, yet 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), it still fits the basic model. This case holds for both our motivating examples above.

We discuss a possible semantic framework to describe the hybrid coupling of the two machine types, and a refinement/retrenchment approach to support a program development approach. Together, these would provide the tools to determine the combined computational power of the heterotic computer.

3.2 Semantic framework

The ultimate goal is a form of *refinement calculus for heterotic computers*, suitable for use by the working programmer. However, producing such a framework first requires theoretical input. In particular, we seek a suitable form of semantics on which the refinement calculus is based. Although such models exist for the individual systems described here, the theoretical challenge is to give a formal description of how such systems may interact in non-trivial ways.

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. Similarly, quantum computation has many models, including the stabiliser formalism, purely category-theoretic approaches, the circuit model, and density matrices particularly suitable for quantum/probabilistic hybrid systems. 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.

These approaches provide background for the development of a semantic framework. 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 $\mathcal{C}_A$ and $\mathcal{C}_B$, we require a formal setting in which both the joint system, and the non-trivial interactions between systems A and B may be modelled. A common approach to modelling a joint system is via the straightforward product category $\mathcal{C}_A \times \mathcal{C}_B$. However, for our purposes, this is entirely unsuitable: the real object of study should instead be the non-trivial interactions between the subsystems. Although *ad hoc* extensions or quotients of the straightforward product category may have some utility, we take a more structural approach, based on the theory of adjunctions [23].

In categorical models of logic and computation, the notion of a monoidal closed category is often fundamental: monoidal closure has a strongly computational interpretation as β-reduction (in Cartesian closed categories) or cut-elimination in logical systems [20], and even compositionality in models of Turing machines [18]. Formally, a monoidal closed category $\mathcal{C}$, has two functors, the *monoidal tensor* $\otimes : \mathcal{C} \times \mathcal{C} \to \mathcal{C}$, and the *internal hom* $[\_ \to \_] : \mathcal{C}^{op} \times \mathcal{C} \to \mathcal{C}$ satisfying various conditions laid out in [23]. In particular, these two functors are related by the following condition

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

This is a canonical example of an adjunction. Further, in the very special case where the system is *untyped* (so all objects of $\mathcal{C}$ are isomorphic), we expect to recover models of *universal computation* (e.g. the C-monoids of [20] or the untyped compact closure of [17]).

For our purposes, monoidal closure, in either its typed or untyped form, is too strong — it is clearly describing the situation where the computation is carried out in a single system. Thus, we take the notion of an adjunction between two functors as primitive, without the additional baggage imposed by categorical closure. The notion of an adjunction is simply a categorification of the concept of a galois connection, thus two functors $\Gamma : \mathcal{C}_A \to \mathcal{C}_B$ and $\Delta : \mathcal{C}_B \to \mathcal{C}_A$. form an adjoint pair when $\mathcal{C}_A(\Gamma(X), Y) \;\cong\; \mathcal{C}_B(X, \Delta(Y))$, for all $X \in Ob(\mathcal{C}_A)$, $Y \in Ob(\mathcal{C}_B)$. 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 symmetry within such interactions.

A concrete example of this notion of adjunction (via its characterisation as unit/co-unit maps in a 2-category setting), used to describe creation of quantum systems from classical data, and measurement of quantum systems (resulting in classical information), can be found in the categorical semantics approach of Abramsky and Coecke [1]. It thus appears that category theory provides ready-made abstract conditions suitable for describing the mutual update of distinct

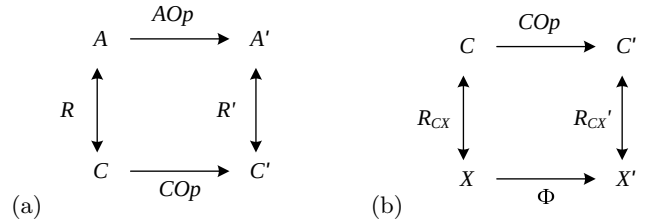

Fig. 3. (a) A simulation, used to prove refinement; (b) Physical and computational layer relationship

systems in heterotic computing, along with real concrete examples of how this works in certain settings.

3.3 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.

Introduction. 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). [5]

[5] Refinement 'reduces non-determinism' until we reach a completely deterministic implementation. It is interesting to interpret this in the case of quantum computation, where the implementation is intrinsically non-deterministic. We classically think of the resolution of non-determinism being under the control of the programmer.

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)[6]. So we can at best require the inexact commutation

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

Analog refinement as Retrenchment? Retrenchment [3–6] 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 [5], 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 [3].

Retrenchment has its critics, but we have argued elsewhere [4] that these criticisms are invalid in the context of real world engineering developments. Here we claim that (a rigorously posed form of) retrenchment is appropriate for casting analog computation in a refinement-like framework. It would be used to analyse the size, nature, and propagation of errors.

Inputs and outputs. 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 [12], 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) [10].

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 our NMR adder [24]: the physical level is the NMR; the computational level is the XOR gate; the initialisation is interpreting a frequency and a phase delay as a bit; the finalisation is observing an integrated signal as a bit.

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

For this form of initialisation/finalisation to work in the analysis, it has to be possible *in principle* to provide all the inputs 'up front', 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 [29]) 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.

Linking outputs to inputs. We have an additional step in the NMR example [24], 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.

Heterotic refinement. The base and controller levels can be implemented ('refined') separately. For analog computing, we expect the base level to be implemented in an analogue medium (NMR, quantum cluster state – hence 'retrenched') and the controller level to be digital, but that is not a necessary restriction.

Example 1: NMR, 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).

Example 2: the quantum cluster state and the parity controller. 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 ('collapses') the state, which is part of the computation.

Such concrete models could 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 [14] — 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 (due to noise, and to 'drift'), and techniques for correction and control of these errors.

4 NMR computing within the framework

NMR technology provides a well-developed experimental system in which a variety of computational implementations can be studied and combined. There is a rich NMR variable space available to be explored, and the interaction between classical and quantum layers in the NMR system can be adapted to many combinations of heterotic compositions. Our existing work [24, 7] on hybrid classical systems provides a good starting point. To illustrate, we outline how the physical NMR system is adapted to perform classical gates, and describe the possibilities for extending this.

4.1 Classical NMR variables

There are three experimental categories of NMR variables that can be used in a computational context: r.f. pulses and pulse sequences; spin system parameters; choice of material/state of condensed matter.

Error compensation in r.f. pulses and pulse sequences. The unwanted effects of imperfect, say π, or $\pi/2$ pulses in NMR experiments have always been at the centre of attention in NMR pulse sequence design. Numerous error compensation techniques exist, most prominently phase cycling is used in almost all circumstances. As the name implies, phase cycling (of transmitter and receiver phases) consists of repeating and co-adding spectra obtained with suitably cycled sets of phases, sometimes up to 64 spectra. This experimental NMR approach to error compensation is acceptable in many analytical NMR applications as poor signal-to-noise ratio makes signal averaging necessary anyway. Here we cannot afford the accumulation of multiple spectra. Instead we need to carefully assess the imperfections in 'single shot' NMR spectra, and determine, for example, the merits of 'composite pulses' [21] in reducing experimental errors.

Characterising spin system parameters. Computational NMR implementations usually require the presence of more than one spin species with different resonance frequencies. The presence of a range of frequencies affects the performance of different pulse sequences in different ways. Some of these effects are experimentally unavoidable, but have an effect in accumulated errors in the results of concatenated pulse sequences. Some pulse sequences that appeal in theory may have to be discarded for practical reasons. The computational-relevant consequences of these effects need to be determined.

Choice of material/state of condensed matter. NMR experiments using isotropic liquids are (usually) straightforward to implement and samples are easy to prepare. But isotropic liquids do not permit the control of resonance frequencies as continuous variables (unless one employs field gradients, which may often not be desirable). Solid state (anisotropic) systems, using either single crystal or the more readily accessible polycrystalline power, can provide this extra variable to be exploited for heterotic computation.

4.2 Analysis of classical and quantum layers in NMR computation

Quantum and classical NMR computational systems can be used as heterotic layers effectively.

The modular NMR tools (pulse sequences, spin systems, state of condensed matter) can be built into specific computational implementations, where they can be analysed to understand and classify the relative merits of classical *vs* quantum NMR implementations. A concrete example is the Deutsch-Jozsa algorithm. Several NMR quantum implementations have been described [9, 11, 22], and one can also think of equivalent classical NMR implementations [25]. Experimental NMR imperfections need to be taken into account, as do specific errors inherent in the classical and/or quantum systems themselves.

The various advantages and disadvantages of classical *vs* quantum NMR implementations with regard to parallelism, superposition, and interference are important. Both classical and quantum implementations of NMR experiments require preparation of the spin system(s) and appropriate read-out procedures. The relative cost of each of these is important in determining how one can exploit the best of both layers. This can be examined by evaluating the performance and preparation needed for a computational task using either a single n-spin system or n single spin species.

5 Stressing the framework with continuous variable computing

The framework described in §3 is a 'stepwise' approach. More complicated heterotic systems will be 'continuing' systems, where the base system's state continues to evolve whilst the control layer computes.

An appropriate experimental basis for studying these more complex heterotic systems is provided by continuous variable computing, both quantum and classical. Existing models of analog computation could be exploited to extend or modify the framework to accommodate continuous variables, or clearly define the framework's applicability as limited to non-evolving base computation.

Despite providing many of the earliest practical computers, analog computation (both classical and quantum) has had much less thorough theoretical development compared to digital computation. Before it can be cast in a heterotic framework, some gaps in the theory would need to be filled. In particular, an approach to analysing the propagation of errors is needed, as is the determinations

of universal gate sets in a substrate-respecting manner (cf [28]). Although underdeveloped for computation, continuous variables (both quantum and classical) are often used in communications channels. It follows that using continuous variables as a control layer in the heterotic computer can take advantage of the more highly developed communications technologies, thus a continuous variable control layer should be easier to achieve than a continuous variable substrate. This architecture is known as a hybrid scheme in quantum computation (eg [26]) and is considered one of the most promising scalable routes to useful quantum computation. The inverse, using a discrete quantum control layer on a continuous quantum base layer, as specified in detail in [28] for a micro-maser experimental system, is suitable for studying continuous variables in the heterotic framework.

6 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, and indicated how classical NMR computation can be advanced to exploit the framework.

This is only the first step in hybrid computation. We have discussed an area that would need enhancement to the framework, where the base layer continues its computation whilst the controlling layer is working. This will be the case for a range of dynamical systems; 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.

References

1. Abramsky, S., Coecke, B.: A categorical semantics of quantum protocols. Proc. IEEE Symp. Logic In Comp. Sci. pp. 415–425 (2004)
2. Anders, J., Browne, D.: Computational power of correlations. Phys. Rev. Lett. 102, 050502 (2009)
3. Banach, R., Jeske, C., Fraser, S., Cross, R., Poppleton, M., Stepney, S., King, S.: Approaching the formal design and development of complex systems: The retrenchment position. In: WSCS, IEEE ICECCS'04 (2004)
4. Banach, R., Jeske, C., Poppleton, M., Stepney, S.: Retrenching the purse. Fundamenta Informaticae 77, 29–69 (2007)
5. Banach, R., Poppleton, M.: Retrenchment: an engineering variation on refinement. In: 2nd Intl. B Conference. LNCS, vol. 1393, pp. 129–147. Springer (1998)

6. Banach, R., Poppleton, M., Jeske, C., Stepney, S.: Engineering and theoretical underpinnings of retrenchment. Sci. Comp. Prog. 67(2-3), 301–329 (2007)
7. Bechmann, M., Sebald, A., Stepney, S.: From binary to continuous gates – and back again. In: ICES 2010. LNCS, vol. 6274, pp. 335–347. Springer (2010)
8. Blakey, E.: Unconventional complexity measures for unconventional computers. Natural Computing (2010), doi:10.1007/s11047-010-9226-9
9. Chuang, I.L., Vandersypen, L.M.K., Zhou, X., Leung, D.W., Lloyd, S.: Experimental realization of a quantum algorithm. Nature 393(6681), 143–146 (1998)
10. Clark, J.A., Stepney, S., Chivers, H.: Breaking the model: finalisation and a taxonomy of security attacks. ENTCS 137(2), 225–242 (2005)
11. Collins, D., et al.: NMR quantum computation with indirectly coupled gates. Phys. Rev. A 62(2), 022304 (2000)
12. Cooper, D., Stepney, S., Woodcock, J.: Derivation of Z refinement proof rules: forwards and backwards rules incorporating input/output refinement. Tech. Rep. YCS-2002-347, Department of Computer Science, University of York (Dec 2002)
13. Cory, D.G., et al.: NMR based quantum information processing: Achievements and prospects. Fortschritte der Physik 48(9–11), 875–907 (2000)
14. d'Hondt, E., Panangaden, P.: Quantum weakest preconditions. Math. Struct. Comp. Sci. 16(3), 429–451 (2006)
15. DiVincenzo, D.P.: The physical implementation of quantum computation. Fortschritte der Physik 48(9–11), 771–783 (2000), arXiv:quant-ph/0002077v3
16. Dusold, S., Sebald, A.: Dipolar recoupling under magic-angle spinning conditions. Annual Reports on NMR Spectroscopy 41, 185–264 (2000)
17. Hines, P.: The categorical theory of self-similarity. Theory and Applications of Categories 6, 33–46 (1999)
18. Hines, P.: A categorical framework for finite state machines. Mathematical Structures in Computer Science 13, 451–480 (2003)
19. Jones, J.A.: NMR quantum computation. Progress in Nuclear Magnetic Resonance Spectroscopy 38(4), 325–360 (2001)
20. Lambek, J., Scott, P.: An introduction to higher-order categorical logic. Cambridge University Press (1986)
21. Levitt, M.H.: Composite pulses. In: Grant, D.M., Harris, R.K. (eds.) Encyclopedia of Nuclear Magnetic Resonance, vol. 2, pp. 1396–1441. Wiley (1996)
22. Linden, N., Barjat, H., Freeman, R.: An implementation of the Deutsch-Jozsa algorithm on a three-qubit NMR quantum computer. Chemical Physics Letters 296(1–2), 61–67 (1998)
23. Mac Lane, S.: Categories for the working mathematician. Springer (1971)
24. Roselló-Merino, M., Bechmann, M., Sebald, A., Stepney, S.: Classical computing in nuclear magnetic resonance. IJUC 6(3–4), 163–195 (2010)
25. Sebald, A., Bechmann, M., Calude, C.S., Abbott, A.A.: NMR-based classical implementation of the de-quantisation of Deutsch's problem, (work in progress)
26. Spiller, T.P., Nemoto, K., Braunstein, S.L., Munro, W.J., van Loock, P., Milburn, G.J.: Quantum computation by communication. New J. Phys. 8(2), 30 (2006)
27. Stepney, S.: The neglected pillar of material computation. Physica D: Nonlinear Phenomena 237(9), 1157–1164 (July 2008)
28. Wagner, R.C., Everitt, M.S., Jones, M.L., Kendon, V.M.: Universal continuous variable quantum computation in the micromaser. In: Unconventional Computation 2010. LNCS, vol. 6079, pp. 152–163. Springer (2010)
29. Wegner, P.: Why interaction is more powerful than algorithms. CACM 40, 80–91 (1997)