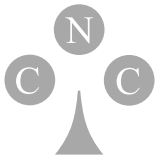
Natural Computing Series

Susan Stepney · Steen Rasmussen
Martyn Amos *Editors*

Computational Matter

Springer

Natural Computing Series



Susan Stepney • Steen Rasmussen • Martyn Amos
Editors

Computational Matter

Springer

Editors
Susan Stepney
Department of Computer Science
University of York
York, UK

Steen Rasmussen
Department of Physics, Chemistry and Pharmacy
FLinT Center, University of Southern Denmark
Odense, Denmark

Martyn Amos
School of Computing, Mathematics
and Digital Technology
Manchester Metropolitan University
Manchester, UK

ISSN 1619-7127
Natural Computing Series
ISBN 978-3-319-65824-7 ISBN 978-3-319-65826-1 (eBook)
https://doi.org/10.1007/978-3-319-65826-1

Library of Congress Control Number: 2018948675

This Springer imprint is published by the registered company Springer Nature Switzerland AG.
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Part II Delving into UCOMP Concepts

List of Contributors

Andrew Adamatzky. Department of Computer Science, University of the West of England, Bristol, UK

Martyn Amos. School of Computing, Mathematics and Digital Technology, Manchester Metropolitan University, UK

Alessandro Bramanti. STMicroelectronics, Lecce, Italy

Hajo Broersma. Faculty of Electrical Engineering, Mathematics and Computer Science, and MESA+ Institute for Nanotechnology, University of Twente, Enschede, The Netherlands

Elisabetta Collini. Department of Chemical Science, University of Padova, Italy

Gianaurelio Cuniberti. Institute for Materials Science, Dresden University of Technology, Germany

Matthew Dale. Department of Computer Science, and York Cross-disciplinary Centre for Systems Analysis, University of York, UK

Paolo Dini. Royal Society Wolfson Biocomputation Research Lab, Centre for Computer Science and Informatics Research, University of Hertfordshire, Hatfield, UK

Victor Erokhin. CNR-IMEM, Parma, Italy

Giuditta Franco. Dipartimento di Informatica, Università di Verona, Italy

Barbara Fresch. Department of Chemical Science, University of Padova, Italy; and Department of Chemistry, University of Liege, Belgium

Angel Goñi-Moreno. School of Computing, Newcastle University, UK

Rafael Gutierrez. Institute for Materials Science, Dresden University of Technology, Germany

Simon J. Hickinbotham. Department of Computer Science, and York Cross-disciplinary Centre for Systems Analysis, University of York, UK

Dawit Hiluf. The Fritz Haber Center for Molecular Dynamics and Institute of Chemistry, The Hebrew University of Jerusalem, Israel

Gábor Horváth. Department of Algebra and Number Theory, Institute of Mathematics, University of Debrecen, Hungary

Jeff Jones. Department of Computer Science, University of the West of England, Bristol, UK

M. V. Klymenko. Department of Chemistry, University of Liege, Belgium

Zoran Konkoli. Department of Microtechnology and Nanoscience, Chalmers University of Technology, Gothenburg, Sweden

Raphael D. Levine. The Fritz Haber Center for Molecular Dynamics and Institute of Chemistry, The Hebrew University of Jerusalem, Israel; and Crump Institute for Molecular Imaging and Department of Molecular and Medical Pharmacology, David Geffen School of Medicine and Department of Chemistry and Biochemistry, University of California, Los Angeles, California, USA.

Vincenzo Manca. Dipartimento di Informatica, Università di Verona, Italy

Richard Mayne. Department of Computer Science, University of the West of England, Bristol, UK

John S. McCaskill. European Center for Living Technology, Ca' Foscari University, Venice, Italy

Julian F. Miller. Department of Electronic Engineering, University of York, UK

Chrystopher L. Nehaniv. Royal Society Wolfson Biocomputation Research Lab, Centre for Computer Science and Informatics Research, University of Hertfordshire, Hatfield, UK

Stefano Nichele. Department of Computer Science, Oslo Metropolitan University, Norway

Steen Rasmussen. Center for Fundamental Living Technology, University of Southern Denmark; and Santa Fe Institute, USA

Françoise Remacle. Department of Chemistry, University of Liege, Belgium

Ross Rinaldi. Department of Mathematics and Physics "E. De Giorgi", University of Salento, Lecce, Italy

Eric Rothstein. Department of Informatics and Mathematics, University of Passau, Germany

Alejandro Santana Bonilla. Institute for Materials Science, Dresden University of Technology, Germany

Daniel Schreckling. Department of Informatics and Mathematics, University of Passau, Germany

Andrew Schumann. University of Information Technology and Management, Rzeszow, Poland

Stefano Siccardi. Department of Computer Science, University of the West of England, Bristol, UK

Susan Stepney. Department of Computer Science, and York Cross-disciplinary Centre for Systems Analysis, University of York, UK

Göran Wendin. Applied Quantum Physics Laboratory, Microtechnology and Nanoscience, Chalmers University of Technology, Gothenburg, Sweden

James Whiting. Department of Engineering, Design and Mathematics, University of the West of England, Bristol, UK

Peter R. Wills. Department of Physics, University of Auckland, New Zealand

Tian-Min Yan. Department of Chemistry, University of Liege, Belgium; and Shanghai Advanced Research Institute, Chinese Academy of Sciences, China

Chapter 1
Introduction

Susan Stepney, Steen Rasmussen, and Martyn Amos

Abstract This chapter provides an introduction to the book *Computational Matter*, providing the historical background to the production of the book, and an overview of its scope and content.

1.1 Historical background

Stanislaw Ulam famously said that "using a term like non-linear science is like referring to the bulk of zoology as the study of non-elephant animals" (Campbell et al., 1985; Gleick, 1987): non-linear science actually forms the bulk of natural science. So we contend it is with *non-standard computing*, also referred to as *unconventional computing* (UCOMP): UCOMP actually forms the bulk of computational science.

UCOMP is the study of computation outside the standard model of conventional computing (which we refer to as *classical computation*, or CCOMP). By 'standard model' we mean the model underlying the implementation of almost all commercially available devices, and where the design of the devices make no special claims about their methods of computation. There is extensive territory outside this standard model, in terms both of theoretical models and of physical implementations, for UCOMP to occupy.

In order to help map out this territory, in 2009 the European Commission's Future and Emerging Technologies agency commissioned and published an Expert Consultation report on *Unconventional Formalisms for Computation* (European Commission, 2009), drafted by one of the editors of this book (SS).

> Conventional (classical) computation may be baldly characterised as that of the Turing/von Neumann paradigm: based on the mathematical abstraction of Turing Machines (or equivalents) with exact provable results, and an implementation in terms of sequential program v data models. It has been incredibly successful. How-

S. Stepney et al. (eds.), *Computational Matter*, Natural Computing Series,
https://doi.org/10.1007/978-3-319-65826-1_1

> ever, there is increasing argument that it encompasses only a small subset of all computational possibilities, and increasing evidence that it is failing to adapt to novel application domains. — (European Commission, 2009)

That report summarised the then-current state of UCOMP research, and made a recommendation:

> The recommendation of this consultation is for a funded programme in **advanced unconventional computation**, intended to take the discipline beyond its current state of relatively isolated individual topics: to fill the gaps, unify areas, break new ground, and build a unified discipline of computation as a whole. Individual projects funded in such a programme should clearly demonstrate how they intend to progress the field as a whole. — (European Commission, 2009)

In 2011 the agency opened the ICT FP7 Call 8 FET Proactive in Unconventional Computation[1], and for associated Coordination Actions. As a result, seven research projects were funded:

- BIOMICS: Biological and Mathematical Basis of Interaction Computing (Dini et al., 2012), www.biomicsproject.eu
- MICREAgents: MIcroscopic Chemically Reactive Electronic Agents (McCaskill et al., 2012), www.micreagents.eu
- MolArNet: Molecular Architectures for QCA-inspired Boolean Networks (Rinaldi et al., 2012), www.molarnet.eu
- Multi: Multi-Valued and Parallel Molecular Logic (Collini et al., 2012), www.multivalued.eu
- NASCENCE: NAnoSCale Engineering for Novel Computation using Evolution (Broersma et al., 2012), www.nascence.eu
- PhyChip: Growing Computers from Slime Mould (Adamatzky et al., 2012), www.phychip.eu
- SYMONE: SYnaptic MOlecular NEtworks for Bio-inspired Information Processing (Wendin et al., 2012), www.symone.eu

One coordination action was also funded:

- TRUCE: Training and Research in Unconventional Computation in Europe (Amos et al., 2012), www.truce-project.eu

The objectives of the TRUCE coordination action were (1) to formulate, develop and maintain a European vision and strategy for UCOMP; (2) to identify areas of importance in UCOMP, and help to focus research in these areas; (3) to provide a framework for the discussion and resolution of current issues in UCOMP; (4) to facilitate improvement in the quality, profile and applicability of European UCOMP research; (5) to encourage and support the involvement of students and early career researchers in UCOMP; (6) to facilitate industrial involvement with UCOMP (Amos et al., 2012).

[1] (Floeck, 2012), cordis.europa.eu/fp7/ict/fet-proactive/calls_en.html#previousfetproactive

This book, edited by TRUCE project leaders, comprises a distilled collection of outputs from the UCOMP projects, including a roadmap for Unconventional Computing (UCOMP) research and development in Europe and beyond, in order to identify new trends, challenges and opportunities.

1.2 Scope and content of the book

1.2.1 Roadmap: Mapping the UCOMP territory

Part I of the book comprises outputs from the TRUCE project's UCOMP roadmap activity. The resulting roadmap has been iteratively developed through information gathering from the literature, from specific experts, and from the wider community. The roadmap provides analyses of several specific UCOMP implementations, leading to a range of recommendations for the future of UCOMP research in Europe and beyond.

The chapters in Part I are:

- Chapter 2. UCOMP Roadmap: Survey, Challenges, Recommendations
- Chapter 3. *In materio* Computation Using Carbon Nanotubes
- Chapter 4. Computing by Non-linear Optical Molecular Response
- Chapter 5. Bioinspired Computing with Synaptic Elements
- Chapter 6. Microscopic Chemically Reactive Electronic Agents
- Chapter 7. Cellular Computing and Synthetic Biology
- Chapter 8. PhyChip: Growing Computers with Slime Mould
- Chapter 9. Decoding Genomic Information

1.2.2 Delving into UCOMP concepts

Part II of the book discusses certain aspects of UCOMP across the breadth of the domain, distilling findings from across the funded UCOMP projects. It covers: philosophy of UCOMP, computability and complexity of UCOMP devices, encoding and representing information in UCOMP devices, a theory of interactive computing, reservoir computing with unconventional material, multivalued logics in nanoscale materials, and molecular automata.

The chapters in Part II are:

- Chapter 10. Philosophy of Computation
- Chapter 11. Computability and Complexity of Unconventional Computing Devices
- Chapter 12. Encoding and Representation of Information Processing in Irregular Computational Matter

- Chapter 13. BIOMICS: A Theory of Interaction Computing
- Chapter 14. Reservoir Computing with Computational Matter
- Chapter 15. Multivalued Logic at the Nanoscale
- Chapter 16. Nanoscale Molecular Automata: From Materials to Architectures

Acknowledgements

We would like to thank our fellow TRUCE project partners, René Doursat and Francisco J. Vico, for their help in making the TRUCE coordination activity such a success, and a pleasure to work on. We thank all the members of the other UCOMP-funded projects, particularly for their enthusiasm for this book: their hard work is what made it possible. And we thank the European Commission, without whose funding the research described herein would not have happened.

We hope that this collected material will be of value to the entire UCOMP community, and more broadly to the computer science and related communities, gathering together common findings from the broadest disciplinary background.

References

Adamatzky, Andrew, Victor Erokhin, Martin Grube, Theresa Schubert, and Andrew Schumann (2012). "Physarum Chip Project: Growing Computers From Slime Mould". *International Journal of Unconventional Computing* 8(4):319–323.

Amos, Martyn, Susan Stepney, Rene Doursat, Francisco J. Vico, and Steen Rasmussen (2012). "TRUCE: A Coordination Action for Unconventional Computation". *International Journal of Unconventional Computing* 8(4) :333–337.

Broersma, Hajo, Faustino Gomez, Julian Miller, Mike Petty, and Gunnar Tufte (2012). "Nascence Project: Nanoscale Engineering for Novel Computation Using Evolution". *International Journal of Unconventional Computing* 8(4):313–317.

Campbell, David, Jim Crutchfield, Doyne Farmer, and Erica Jen (1985). "Experimental mathematics: the role of computation in nonlinear science". *Commun. Assoc. Comput. Mach.* 28:374–384.

Collini, E., R.D. Levine, F. Remacle, S. Rogge, and I. Willner (2012). "Multi Project: Multi-Valued and Parallel Molecular Logic". *International Journal of Unconventional Computing* 8(4):307–312.

Dini, Paolo, Chrystopher L. Nehaniv, Attila Egr-Nagy, Maria J. Schilstra, Daniel Schreckling, Joachim Posegga, Gabor Horvath, and Alastair J. Munro (2012). "BIOMICS Project: Biological and Mathematical Basis of Interaction Computing". *International Journal of Unconventional Computing* 8(4):283–287.

European Commission (2009). "Unconventional Formalisms for Computation". *Expert Consultation Workshop*.

Floeck, Dagmar (2012). "European Support for Unconventional Computation". *International Journal of Unconventional Computing* 8(4):281–282.

Gleick, James (1987). *Chaos: Making a new science*. Viking Penguin.

McCaskill, John S., Gunter von Kiedrowski, Jurgen Ohm, Pierre Mayr, Lee Cronin, Itamar Willner, Andreas Herrmann, Steen Rasmussen, Frantisek Stepanek, Norman H. Packard, and Peter R. Wills (2012). "Microscale Chemically Reactive Electronic Agents". *International Journal of Unconventional Computing* 8(4):289–299.

Rinaldi, R., G. Maruccio, V. Arima, G.P. Spada, P. Samori, G. Cuniberti, J. Boland, and A.P. Bramanti (2012). "Molarnet Project: Molecular Architectures for QCA-Inspired Boolean Networks". *International Journal of Unconventional Computing* 8(4):289–299.

Wendin, G., D. Vuillaume, M. Calame, S. Yitschaik, C. Gamrat, G. Cuniberti, and V. Beiu (2012). "SYMONE Project: Synaptic Molecular Networks for Bio-Inspired Information Processing". *International Journal of Unconventional Computing* 8(4):325–332.

Part I
Mapping the UCOMP Territory

Chapter 2
UCOMP Roadmap: Survey, Challenges, Recommendations

Susan Stepney and Simon J. Hickinbotham

Abstract We report on the findings of the EU TRUCE project UCOMP roadmap activity. We categorise UCOMP along hardware and software axes, and provide a catalogue of hw-UCOMP and sw-UCOMP examples. We describe the approach we have taken to compiling the roadmap case studies in the remainder of Part I. We identify eight aspects of UCOMP to analyse the broad range of approaches: speed, resource, quality, embeddedness, formalism, programmability, applications, and philosophy. We document a range of challenges facing UCOMP research, covering hardware design and manufacture, software theory and programming, and applications and deployment. Finally, we list some recommendations for focussing future UCOMP research: scaling up hardware, software, and concepts; identifying killer apps; and unifying the UCOMP domain.

2.1 The EU TRUCE project roadmap

The EU TRUCE coordination action project (Amos et al., 2012) includes a task of scientific research roadmapping, with a final deliverable comprising a set of case studies, a survey of the domain, the challenges facing UCOMP, and a set of recommendations. The survey, challenges, and recommendations are summarised in this chapter; the case studies comprise the remaining chapters of Part I.

2.2 An atlas, not a roadmap

Unconventional computing (UCOMP) is a fundamental scientific domain, and there are challenges in mapping its scope and development. The act of

S. Stepney et al. (eds.), *Computational Matter*, Natural Computing Series,
https://doi.org/10.1007/978-3-319-65826-1_2

roadmapping such an emerging research domain contrasts with conventional technology roadmapping focussing on a specific technological need:

> Technology roadmapping is driven by a need, not a solution. It is a fundamentally different approach to start with a solution and look for needs. Technology roadmapping provides a way to identify, evaluate, and select technology alternatives that can be used to satisfy the need. (Garcia and Bray, 1997)

The UCOMP domain certainly encompasses and is driving many novel and exciting implementation technologies, but these cover only a part of the possible domain, not its entirety. Nevertheless, it is still necessary to describe and plan the current scope of development of UCOMP, in order to set out a work programme for the coming decades.

In order to distinguish this work from traditional technological roadmapping, we use the term *atlas* as a more appropriate term. A roadmap focusses on getting to a specific destination; an atlas maps out a whole terrain. 'Atlas' raises the idea of land masses, peninsulas, isolated islands, oceans, and uncharted regions. The distribution of research in UCOMP can be described via analogy with such an atlas.

In our UCOMP atlas, an 'ocean' may be an area of little interest in itself, that merely acts as a measure of the distance between land masses. Alternatively, it may be a region of theory which links together otherwise dissimilar UCOMP practices. Either way, it is important to declare whether these oceans have been charted sufficiently well, since there is a possibility that we might miss a significant technique otherwise.

There are two important 'continents' in UCOMP, namely Quantum Computing and ChemBioIT, each of which has recently had roadmaps commissioned and published (Calarco et al., 2010; Hughes et al., 2004; McCaskill et al., 2014), setting out the future of their topic over the next ten years or so.

Taking the analogy further, we might consider how CCOMP sits on the landscape just described. We consider CCOMP as the spread of *computational civilization* across the UCOMP landscape, exploiting resources and enriching its capabilities as it goes.

The TRUCE UCOMP research atlas thus provides (requirements adapted from Sloman (2007)):

- A specification of the problem domain (scope: the atlas as a whole)
- A breakdown into subdomains (structure: the macro-level organisation of the atlas)
- Potential routes to populating the domain (projects: case studies that illustrate the current and future range and status of UCOMP)

In this chapter we give a categorisation of the entire UCOMP landscape extracted from the literature and from a practitioner questionnaire (Stepney and Hickinbotham, 2015, App. A), and we discuss eight aspects of UCOMP that we use to analyse the case studies. We summarise the current state of

	hw-CCOMP	hw-UCOMP
sw-CCOMP	*CCOMP* • some simulations of UCOMP	*hw-only UCOMP* : eg • logic gates and TMs in unconventional substrates
sw-UCOMP	*sw-only UCOMP* : eg • some simulations of full UCOMP • AI/ALife algorithms	*full UCOMP* : eg • analogue computers • quantum computers • (postulated) hypercomputers • embodied unconventional algorithms

Fig. 2.1 UCOMP classified by whether it is hw-only UCOMP, sw-only UCOMP, or full UCOMP. Note that some UCOMP may be simulated in CCOMP.

UCOMP in terms of this classification, based on analysis of the detailed case studies. We provide some recommendations and guidance on where UCOMP research and development should be headed, based on analysis of the detailed case studies.

In the rest of Part I we present the contributed case studies.

2.3 Hardware-UCOMP and software-UCOMP

The simplest axes of classification from which we start are the conventional ones of *hardware* and *software*.

The *hardware* axis captures the physical or material substrate, suitably engineered, that is performing the computation. In CCOMP this is traditional silicon chips (substrate) with a von Neumann architecture (engineered configuration).

The *software* axis captures the abstract or mathematical model of the desired computation, and model of the hardware configuration needed to achieve this. The configuration includes a combination of both program and hardware architecture model; we do not explicitly separate these, since some unconventional approaches combine these components. In CCOMP this is the traditional Turing machine and Boolean logic model.

We classify the COMP landscape coarsely into four regions, depending on whether the hardware, or software, or both, are UCOMP, and give some typical examples, in Figure 2.1.

The boundary between UCOMP and CCOMP is diffuse and movable. Where do we draw the CCOMP/UCOMP line along the spectrum of single core through multi-core to massively parallel hardware (let alone exotic substrates)? Where do we draw it along the spectrum of stand-alone through real time to embedded software applications? However, here we are not con-

cerned with this boundary region so much as the extremes of the spectrum, so arguments about its precise location is not important for our analysis.

Some, but by no means all, members of the UCOMP community require the substrate to be unconventional: they do not recognise sw-only UCOMP; some others recognise only full UCOMP (Stepney and Hickinbotham, 2015, A.2.1) Here we consider all three possibilities (sw-only UCOMP, hw-only UCOMP, and full UCOMP), as each has its interesting properties.

2.3.1 hw-UCOMP: unconventional substrates

Our classification of hw-UCOMP substrates is organised in the hierarchy below. The first level of the hierarchy identifies traditional academic disciplines: Engineering; Physics; Chemistry; Biochemistry; Biology. The next levels identify specific example substrates within that discipline. We identify some examples at this level, mentioned in the roadmap questionnaire responses, to indicate the diversity that exists; there is no attempt to be exhaustive, as novel computational materials are ever being posited and examined.

2.3.1.1 Engineering

Silicon transistors are used in the single classical chip set, and also in a variety of approaches to parallelism: massive multi-core systems, general purpose graphics processing (GPGPU), and field programmable gate arrays (FPGA).

More unconventional devices can be engineered at the nanoscale (the scale of atoms and molecules). Examples include superconducting qubit arrays, memristor arrays, and atomic switch networks.

A variety of media can be engineered to provide general purpose analogue computing. These include field programmable analogue arrays (FPAA), Łukasiewicz Logic Arrays (Mills et al., 1990), and Mills' 'foam' substrates (Mills et al., 2006).

There is a rich history of special purpose mechanical analogue devices, crafted to perform specific computations. These include such diverse devices as the orrery for planetary motions, Kelvin's tide-predictor, early curve integrators, rod and gear integrators, Babbage's clockwork difference and analytical engine, and the hydraulic MONIAC (Monetary National Income Analogue Computer).

2.3.1.2 Physics

A variety of physical processes are exploited for UCOMP.

Josephson junction circuits are used in quantum computing. Quantum dots, as well as being exploited in quantum computing, can be used for programmable matter.

Optical devices are used in both classical optical computers and in quantum-optics computers that exploit photons as qubits.

The technology of Nuclear Magnetic Resonance (NMR), which combines properties of nuclear spins, magnetic fields, and radio frequency pulses, can be used to perform classical computing (Roselló-Merino et al., 2010) and quantum computing (Gershenfeld and Chuang, 1997).

Complex physical materials have been repurposed for computation in several cases. These include liquid crystal (Harding and Miller, 2004; Harding and Miller, 2005; Harding and Miller, 2007), carbon nanotubes and amorphous substrates (Clegg et al., 2014), graphene, and non-linear magnetic materials.

2.3.1.3 Chemistry

In chemical computing, the computation exploits the chemical properties and reactions of the substrate.

Examples of chemical computing include chemical variants of field computing, and a range of reaction-diffusion systems (Adamatzky et al., 2005) such as Belousov–Zhabotinsky (B-Z) reaction systems (Stovold and O'Keefe, 2017).

Another popular form of chemical computing is droplet systems. The droplets themselves may be the units of computation, may package up and transport chemicals, and may host and interface local B-Z and other reaction-diffusion systems.

2.3.1.4 Biochemistry

Biomolecules including DNA and RNA can be induced to perform computation in a range of approaches.

The most obvious approach is to exploit DNA/RNA as an information-carrying material, and use this to program generic logic circuits (Carell, 2011) using gene expression.

An alternative approach is to exploit the stiffness of short strands of DNA as a construction material, and use this in concert with information carrying 'sticky ends' to self-assemble structures (Rothemund et al., 2004) using tiling theory.

2.3.1.5 Biology

Biology provides a very rich set of systems that can either be used directly for computation, or can provide inspiration for novel computational systems.

Biological systems and subsystems used in this way include neurons, plasmids, the immune systems, evolution, viruses, phages, single bacteria, synthetic biology, communicating bacteria, bacterial mats, mammalian cells, and slime moulds.

2.3.1.6 Hybrid systems that combine/compose more than one substrate

As with any engineering discipline, hw-UCOMP may combine a range of substrates into a computational system. Examples include: a conventional computer plus a UCOMP "co-processor"; electronics combined with chemicals; neurons attached to silicon; droplets interfacing reaction-diffusion chemicals; chemicals and bacteria in chemotaxis systems; optical and molecular systems; optical and bacterial systems engaged in phototaxis.

2.3.1.7 Other

There are several theoretical or conceptual substrates, which are used to illustrate an unconventional principle, and to explore limits, but are not implemented, beyond some toy cases. Such systems include spaghetti sorting (Dewdney, 1984); gravity sorting (Arulanandham et al., 2002); prisms for sorting with light frequencies (Schultes, 2006); soap film minimisation computers (Isenberg, 1976); elastic band convex hull computers (Dewdney, 1984); billiard ball computers (Fredkin and Toffoli, 1982); conjectured systems such as black holes and other exotic space-times (Earman and Norton, 1993).

2.3.2 sw-UCOMP: unconventional models

We provide the following classification of sw-UCOMP. Note that some examples can occur in more than one category. Some of these models are primarily theoretical, used to investigate aspects of computability and complexity. Others may be executed in CCOMP simulation (potentially with loss of efficiency), or may be executed in unconventional substrates to give full UCOMP.

Quantum models include: quantum circuit models; quantum annealing models, including adiabatic computing; quantum information; quantum communication.

Continuum models, and massively-parallel/spatial models, include: classical analogue computing; quantum analogue computing; field computing; ubiquitous computing; cellular automata.

Bio-inspired models include: population-based search algorithms (evolutionary, immune, swarm, etc); computational morphogenesis and developmental models; membrane-based models (brane-calculi, P-systems, etc); neural models (ANNs, reservoir computing); gene regulatory network models; metabolic network models; Artificial Life models (ALife).

Growth and self-assembly models include: Artificial Chemistries (AChems) and reaction networks; L-systems and other generative grammars; computational morphogenesis and developmental models; dynamical systems models, including reservoir computing; tiling models; self-modifying models.

Non-halting models include: process algebras; interaction computing.

Hypercomputing models include: Newtonian physics models; general relativistic models; infinite precision real number models.

2.3.3 Full UCOMP: unconventional models in unconventional substrates

Many of the unconventional substrates have been used to implement small CCOMP computations: logic gates or small circuits. Example systems that use unconventional substrates to implement sw-UCOMP models, to yield full UCOMP, include:

- analogue computers
- quantum computers, quantum communications
- evolution *in materio* computing
- reservoir computing *in materio*
- tile assembly of DNA tiles

Often there needs to be a careful choice of substrate and model, so that the computational model can exploit the properties of the substrate to perform its computation effectively.

2.3.4 Simulating UCOMP

Some practitioners require that true UCOMP not be efficiently simulable in CCOMP (to be super-Turing), but this seems overly restrictive: for one thing, it assumes a Turing model of computation; on the other hand it excludes simple logic circuits implemented in cells using synthetic biology.

We can categorise UCOMP approaches based on their simulability in CCOMP algorithms on CCOMP hardware as follows:

- Efficiently simulable models, possibly with a transfer of space resources in the model to time resources in simulation: for example, cellular automata.
- Super-Turing models, simulable in CCOMP, but not efficiently: for example, some quantum computing algorithms.
- Hypercomputing models, uncomputable in CCOMP and hence not simulable. Whether such models are physically implementable in UCOMP substrates is contentious (see Chapter 11).

Simulation is an important technology in UCOMP research, where expensive and specialised substrates can be difficult to obtain. Simulation allows UCOMP models to be explored and tested on conventional machines. Care needs to be taken when drawing detailed conclusions that may be true in simulation, but not carry over to physical substrates.

In addition to CCOMP clusters, silicon technologies such as GPUs and FPGAs may be suitable testbeds for simulating many UCOMP models and substrates, due to their potential for massively parallel computations.

2.4 Eight aspects of UCOMP

The roadmap questionnaire responses (Stepney and Hickinbotham, 2015, App. A), which fed into the above classification, reveal that UCOMP is a field that will continue to be dynamic for at least the next 25 years, and beyond. Current topics will either become accepted or discarded. New techniques will arise at the limits of what is conventionally possible or imaginable. Given this flux, the act of mapping UCOMP becomes more difficult, if we were to focus on particular techniques. Instead, we identify a set of computational properties, which we call *aspects*, and appraise the current and future development of particular techniques in light of these aspects.

Based on the roadmap questionnaire responses, and information from a literature review, and following on from the domain classification, we have identified eight aspects of UCOMP. Our goal was to select aspects that capture the range of potential of the entire UCOMP canon, which are in some sense independent of each other, and for which it is possible to get consensus on the potential of a UCOMP topic. Here we describe these eight aspects, and how they play out across the various UCOMP areas.

These aspects can be thought of as analogous to the DiVincenzo *promise criteria* used in the quantum computing roadmap (Hughes et al., 2004). The UCOMP aspects are:

1. Speed
2. Resource
3. Quality
4. Embeddedness

5. Formalism
6. Programmability
7. Applications
8. Philosophy

One of the uses for this atlas is to indicate the research areas which show the most promise as an augmentation of CCOMP, so UCOMP technologies should be considered on a scale relative to the current and 'best estimate' future state of CCOMP.

For each aspect, we have devised one or more 'prompting questions', which are intended to help the responders compile their case study descriptions consistently. One danger to be wary of is applying well-known conventional or practical CCOMP thinking to the UCOMP paradigm, and thereby missing the potential for a different way of thinking about the process of computation. We point out these risks as we describe each aspect.

Here, we describe each aspect, we mention how the aspect appears in CCOMP, and we summarise the UCOMP results as given in the case studies discussed in more detail in the following chapters.

2.4.1 Speed

Prompting question

Including programming and staging time, how fast is computation in your UCOMP research?

When assessing the utility of any computation, a key consideration is how much time it will take to obtain the output. In CCOMP, speed is usually a direct measure of the rate of processing. We can usually measure the absolute 'wall clock' time it takes to perform a computation, or we can measure the number of steps an algorithm takes and so obtain a speed estimate that is independent of the physical processor speed.

Other issues also need to be considered. There may be a trade-off between speed and accuracy for example. There may be no identifiable steps to count.

Speed is not merely related to the execution time of a UCOMP program. Because of the nature of the different strands of UCOMP, the execution time is only one part of the wider computational process. Writing or creating a computational algorithm can be a challenging process in itself. Setting up the algorithm, inputting the data, and extracting the outputs of the algorithm may be non-trivial in some applications. All of these should be considered when assessing the speed of a particular technique.

Given these observations, it becomes difficult to define what the potential offering of speed is for some UCOMP topics in a manner that allows

direct comparison. Wherever difficulties arise, it should always be possible to measure the time between input and output. This time should then be used as the basis of a comparison of the merits of the technique and a CCOMP equivalent.

CCOMP

There is evidence that conventional chips have reached the limit of miniaturisation and the speed of processing may not improve much without replacing the current technology (Haron and Hamdioui, 2008). Assuming that it is not possible to simultaneously replace current transistorisation technology and increase processor speed, it is reasonable to suggest that processing speed in CCOMP will stay constant.

UCOMP

- There is a huge variation in 'clock speed' across materials, from femtosecond (10^{-15} s, physics/optical) and nanosecond (10^{-9} s, physics/electronic, logic gates and memristors), to seconds (chemistry), to minutes/hours (biology).
- In some cases with non-clocked models of computation (including certain analogue, continuous time evolution, and relaxation approaches), the effective speed may be unknown, and claims may be controversial.
- Most approaches benefit from massive parallelism.
- For embedded systems, the computation speed relative to the controlled system's timescales is the important factor. Absolute speed, or speed in comparison to CCOMP, is not the relevant metric.
- For hybrid systems, the relative computation speeds of the different substrates is an important factor; where they are very different, matching issues may exist.
- In some approaches, the hardware 'decays' (from quantum decoherence to biocell death), and so the important measure is how much computation can be done before the hardware is no longer usable.
- There are different aspects of speed:
  - hardware construction timescale: in CCOMP this is neglected, as the hardware is used repeatedly. However, this timescale is particularly significant in UCOMP for 'one-shot' devices where the computation destroys or consumes the device.
  - program loading timescale: in CCOMP this is generally neglected; in the Turing model the program is present on the tape at the start of the computation. However, this timescale can be significant in UCOMP if program loading corresponds to some slow configuration of the

substrate, or some slow search for the required configuration (as in *in materio* computing).
– input/output timescale: in CCOMP this is generally neglected; in the Turing model the input is present on the tape at the start of the computation, and the computation halts leaving the output present on the tape at the end. Communication applications are more concerned with I/O timescales, and transmission timescales; embedded applications with analogue-to-digital and digital-to-analogue conversion rates. These timescales can be significant in UCOMP if I/O requires significant data configuration and conversion.

2.4.2 Resource

Prompting question

How does the economic cost of the computational resource compare to the cost in CCOMP?

This aspect concerns the consumption of physical resource by the UCOMP process. In CCOMP where the main consumable is electrical power, this is an active area of research (ITRS, 2013). There is a wider range of resource use in UCOMP (Blakey, 2017). A benchmark of some utility is the economic cost of the resource: how easy is it to procure the resource, what are the by-products, and how easy is it to dispose of them. This allows the resource usage in a slime-mould computation, say, to be compared with electrical power consumption in CCOMP in a more objective manner.

Another facet of the resource aspect concerns the *scalability* of the technology. Costs should be considered per unit of computational resource, and by the total computational resource of however many units can reasonably be composed.

CCOMP

It is possible that processors which use less power than currently will be developed, whilst maintaining the symbolic nature of CCOMP.

UCOMP

- Most equipment is still highly specialist and expensive; but it is early days, and prices will come down.

- Some systems require expensive and power-hungry lasers; however research may be able to engineer some of these to run off less power hungry sources, and even sunlight instead.
- Some DNA-based systems require complex and slow laboratory synthesis processes; however, these resources will reduce as technology improves.
- Memristor memory may eventually be significantly cheaper than DRAM. Architectures based on data processing at the memristor memory level, avoiding processor-storage bottlenecks, have the potential to be much more energy efficient and faster than traditional CCOMP implementations.
- Some areas are non-comparable, such as synthetic biology, where there is not a meaningful CCOMP equivalent implementation.

2.4.3 Quality

Prompting question

Does your UCOMP research offer qualitatively different computation to CCOMP?

Quality is concerned with the manner in which a computation is carried out: precision; fit to underlying model (if the model is known); richness; stochasticity; repeatability. The quality of a UCOMP method concerns its potential to offer an analysis of the inputs that is different from a CCOMP analysis. For example, one could consider how the workings of an analogue computer can yield different results from a digital computer by virtue of its numerically continuous internal representation of the problem. The output might be a state of a physical system that could be interrogated at many different scales.

CCOMP

Since much of the definition of quality is relative to CCOMP, quality cannot progress much here. There is a potential for simulations on CCOMP platforms to emulate some unconventional processes.

UCOMP

- There are many novel models, offering benefits over, and differences from, CCOMP:

  - various quantum models, with potential exponential speedup for a limited class of problems
  - inherent massive parallelism, trading off time against space, avoiding traditional von Neumann bottlenecks
  - analogue / continuous models, with continuous processing, and continuous input/ouptut
  - 'sloppy', fuzzy, and probabilistic models, promising faster execution, but without a guaranteed or repeatable result
  - stateful logic, e.g., memristors
  - non-binary, and non-digital, encodings
- Some UCOMP substrates might not have any associated well-defined computational model; they are still being investigated for their potential. Search can be used to program devices despite there being no underlying model.
- Fit to the associated computational model varies across a spectrum. Physical hardware (especially quantum) tends to fit the associated model more closely than does biological wetware (where the intrinsic variability and noisiness of the wetware is often abstracted away in the associated model, despite the fact that biological systems may have evolved to exploit noise).
- An important difference is in hybrid devices exploiting multiple UCOMP substrates, and multiscale structures.
- There are issues around input and output for unconventional models; each might require significant data configuration and conversion, which might involve significant 'hidden' computational requirements.

2.4.4 Embeddedness

Prompting question

Can your UCOMP research be implemented within the process it is aimed at controlling/monitoring?

Embeddedness concerns the relationship between the performance of a computation, its output, and what the output is used for. How much does the computation rely directly on physical characteristics of the substrate, rather than only through an abstraction layer? Is the output useless unless it controls a system directly, rather than being abstractly read off and then fed into a system?

CCOMP

Embeddedness is never going to be high for CCOMP due to the highly abstract symbolic processing of data that it employs.

UCOMP

- Not all UCOMP devices are embedded; some are alternatives to standalone CCOMP devices.
- Even standalone unconventional substrates often have special purpose interface devices to link them to CCOMP kit: they are usually hybrid devices of UCOMP+CCOMP.
- For some UCOMP devices, being embedded is the *raison d'être* of the substrate choice; this tends to be certain chemical and biological devices designed to be embedded/embodied in chemical/biological systems.
- Embedded systems may suffer from the 'wiring problem', of moving data from one part of the processing to another – for example, signalling within or between cells – in a controlled and effective manner.
- Embeddedness can remove some of the input/output issues, by allowing a closer match between data formats in the controller and controlled system (cf the need for analogue-to-digital and digital-to-analogue converters in many embedded CCOMP devices). However, there is the complementary issue of programming the devices, which also needs I/O.

2.4.5 Programmability

Prompting question

How easy is it to write sophisticated programs in the computational substrate that you study?

This aspect concerns how UCOMP systems may be programmed. The ubiquity of CCOMP masks the challenges involved in organising an algorithm on a substrate. For example, in computation *in materio*, it is suggested that evolutionary search is the principal means of 'programming' the substrate (Miller et al., 2014).

CCOMP

This is well understood and there are myriad ways of writing, compiling and executing programs.

UCOMP

- There are existing quantum computing languages, both for digital and analogue devices.
- Some UCOMP systems use CCOMP-like approaches, such as gates and circuits.
- There are some multi-paradigm approaches for hybrid devices and multi-scale devices.
- In synthetic biology, there are high-level programming languages that 'compile' down to low-level components such as gene sequences, searching through databases to find suitable combinations of implementations.
- Evolution/search/learning is used where there is no good understanding or underlying model of the substrate (for example, *in materio* computing), and for non-modular complex emergent properties where classical decomposition approaches cannot be used.
- In some systems, the program is part of the substrate design: the system is application-specific, rather than reprogrammable.

2.4.6 Formalism

Prompting question

Does your UCOMP research topic have a formalism that augments/extends CCOMP formalism?

CCOMP systems have many formalisms that can be used to design algorithms etc. regardless of their hardware implementation. Few UCOMP systems have such a formalism, and most exist outside the standard Turing Machine model that underpins the formalisms of CCOMP.

Formalisms allow us to understand in abstract terms the expressiveness of a system, its limitations, and its relation to other systems and CCOMP. Whilst an over-reaching formalism for UCOMP may be desirable, it is equally desirable that UCOMP researchers foster development of formalisms for their own particular techniques.

CCOMP

Since CCOMP is purely symbolic and the underlying philosophy is well connected to the methods of implementation, it is reasonable to suggest that a rounded formalism exists for CCOMP.