Proposed subtopic for GC7: "Journeys in Non-Classical Computing"

Nanotechnology: opportunities and challenges

Robin Milner, The Computer Laboratory, University of Cambridge
Susan Stepney, Department of Computer Science, University of York

Summary

Nanotechnology presents research challenges that will lead to a greatly enriched and more general science of computation. Safety and dependability will present unprecedented demands; the science will be responsible not only for robust design to meet these demands, but for robust analysis that shows they have been met.

1 Background and context

Nanotechnology is the design, development and use of devices on the nanometre (atomic) scale. Here we are not so much concerned with nano-scale artefacts that take the current trend of miniaturisation a few orders of magnitude further. Rather we are interested in *active* physical nano-devices that themselves manipulate the world at their nano-scale in order to manufacture *macroscopic* artefacts. This is Drexler's [D86][D92] vision of nano-scale *assemblers* that build (assemble) macroscopic artefacts. (Such assemblers are often known as *nanites* or *nanobots*.)

In order for nanites to build macroscopic objects in useful timescales, there needs to be a vast number of them. A starting population of a few nanites assembles more of their kind, which then assemble more, with exponentially growing numbers. Once they exist in sufficient numbers, they can build, or become, the macroscopic artefact. This view of nanotechnology promises many awe-inspiring possibilities.

Some argue that such a technology is too good to be true, or at least question the detail of Drexler's predictions. But one should note that there is no conclusive counter-argument to them; indeed, proteins and their associated cellular machinery routinely assemble macroscopic artefacts, or, to use more biological terminology, they *grow organisms*. Here we discuss computational structures that will be relevant whenever *some* technology for sophisticated populations of nanites is achieved, even if not all that has been predicted.

In principle it is possible for nanites to assemble *any* physical artefact, by carefully controlled placement of every component atom (possibly requiring the use of much scaffolding). But in general this is infeasible: in the worst case it could need the *global* control and choreography of the behaviour of every individual nanite. A more feasible approach is to exploit mainly *local* cooperation between suitably-programmed neighbouring nanites, possibly mediated by their shared local environment (which also more closely mirrors the way biological organisms grow).

In order for nanotechnology to be possible, the initial nanites must be fabricated somehow. This complex engineering problem requires collaborative research by physicists, chemists, engineers, and biologists. To the extent that the nanites need to be *programmed* to perform their assembly tasks, computer science (CS) also has a crucial role. We need to develop capabilities to design, program and control complex networks of nanites, so that they safely and dependably build the desired artefacts, and so that they do *not* accidentally build *un*desired ones.

Initial CS research needs to focus on potential ways of designing and assembling artefacts in ways that can be described in terms of predominately local interactions, that is, in terms of the *emergent properties of vast numbers of cooperating nanites*. This requires *analysis* of emergent behaviour; given the orders of magnitude involved, this can be done only with a hierarchy of computational models, explaining the assembly at many different levels of abstraction.

2 Required CS advances

What CS theory and practice do we need in order to be able to design, program and control networks of nanites?

Emergent properties

We need a pragmatic theory of *emergent properties*.

In much the same way that an organism is an emergent property of its genes and proteins (and more), the assembled artefact will be an emergent property of the assembling nanites and their programming. In general, this problem is computationally irreducible, that is, there are no "short cuts" to understanding or prediction, beyond watching the behaviour unfold. Thus reasoning about the precise behaviour of arbitrary networks with a number of nodes comparable to the number of cells in the human body ($\sim 10^{13}$) is (currently) well beyond the state of the art. However, inability to solve the general problem, in principle or in practice, does not prevent exploration of large classes of specific and interesting problems. So we merely need a *sufficient* theory, one that enables us to design nanites to build the many artefacts of interest, and to analyse them for safety and dependability. Certain classes of useful emergent properties may well be tractable to reasoning. For example, many organisms contain emergent hierarchical branching structures, such as arteries, lungs, nervous systems, and, of course, prototypical tree branches. Such emergent structures are particularly straightforward to "program", as evidenced by L-systems [PrL].

Growth and Development

We need a pragmatic theory of *development and growth*.

A population of nanites first "grows" a vastly larger population, then "grows" the artefact in question. Again, we need a *sufficient* theory of growth – to enable us to reason about structures that are the result of a growth process.

Biological insights from embryology and development will be fruitful here, and the relevant ideas need to be abstracted and adapted for nanite assemblers. This "artificial development" also has its own properties: for example, the use of scaffolding will probably be much more important.

Which features of biological organisms are consequences of growth in general, and which are consequences of "wet-ware" growth, and so are different in assembled hardware? What constraints are there in the growth process: is it possible to "grow" a *cooked* steak *ab initio*, or must if first be grown raw (isolated, or as part of a cow), and then chemically modified?

Complex Networks

We need a pragmatic theory of *dynamic, heterogeneous, unstructured, open networks* [Ste]. This topic is already covered by the existing GC7 Journey: *Non-Classical Interactivity – Open Dynamical Networks.*

Complex Adaptive Systems

All these CS advances mentioned above would have application well beyond nanotechnology. All are basic requirements for the general area of *Complex Adaptive Systems*, of which nanotechnology is but one exemplar. Real world examples of CASs include swarms and flocks, ants, immune systems, brains, autocatalytic networks, life, ecologies, and so on. Artificial CASs include complex control systems (industrial plants, Air Traffic Control, etc), eCommerce supply chains and webs, telecoms systems and the Internet, and ubiquitous computing with its hordes of communicating smart devices, economic systems, and so on.

3 Behavioural Modelling

The pragmatic theories for Complex Adaptive Systems, above, must be developed in response to the challenge of nanotechnology, but they need not start from scratch. During the last two or three decades computer scientists have eroded the boundary between programming, which *prescribes* behaviour of a system, and modelling, which *analyses* it. This trend arises naturally from a change of emphasis, from stand-alone computers doing one thing at a time to *distributed* systems – networks of devices each acting independently, with no centralised control. The resulting computational models are in varying degrees logical, algebraic, non-deterministic, stochastic. They have been effectively used to analyse programming languages and communication disciplines. They have also been applied to computer security, mobile phone systems, behaviour in ant colonies, business processes, and signal transduction in biological cells.

A large system such as the Internet can be modelled at many levels of *abstraction*, correlated where possible with the structure of the system. At the higher levels, the analysis of agents' behaviour need not depend on the underlying technology used to realise them. A natural research direction is therefore to extrapolate existing CS models to nanosystems where, despite orders of magnitude increase in population size (compared with, say, the Internet), many of the same general principles of emergence and behaviour should apply.

At the lowest levels of abstraction, which may be called *embodiment*, the analysis of agents' behaviour depends crucially the underlying technology used to realise them. For example, individual nanites are made of only small numbers of atoms, so a one-atom mutation to a nanite – caused by faults in manufacture, by other nanites, by random impact of cosmic rays – could have a dramatic effect on behaviour. In order to reason about the kinds of changes that mutations might make (to reason about the "adjacent possible" [Kau] of the nanite), it is essential to know the detailed make-up and characteristics of the system undergoing mutation.

Close cooperation is therefore needed among many research disciplines, of which CS is one, in order to understand nanopopulations fully. From the CS viewpoint, the gain will be a greatly enriched and more general science of computation.

We continue this section by summarising some of the concepts, theories and tools that CS can bring to the cooperation at the outset. We cite only a selection from the large literature.

Stand-alone computation

Before distributed computing systems became the norm, much computing research laid foundations for the models and tools that those systems need. A start was made in establishing the *verification* of computer programs as an activity in formal logic [Flo][Hoa]. Tools for computer-assisted verification, especially for computer hardware designs [Gor], were pioneered. The status of computer programs as mathematical descriptions of behaviour was established [ScS]. Theories of types began to emerge as a powerful aid to behavioural analysis as well as to programming [Rey]. Even in the 1940s, von Neumann's model of self-reproducing cellular automata anticipated some of the central ideas of nanotechnology [voN].

Abstract machines and process calculi

The first model to capture the complex interplay between non-determinism and concurrency in distributed systems was Petri Nets [Pet], these nets were designed information flow in natural as well as man-made systems. In the early eighties, algebraic process calculi [BHR][Mil] were designed to model interactive systems hierarchically, and to model their behaviour abstractly. The Chemical Abstract Machine [BeB] captured the spatial structure of systems. The pi calculus [MPW] and mobile ambient calculus [CaG] made a further step in modelling systems that can reconfigure both their spatial arrangement and their connectivity.

These models have influenced the design of programming and specification languages, for example LOTOS, occam and Handel-C, and Ada. They have been developed to model systems stochastically, and to deal with hybrid discrete/continuous systems. Recently their theory has been seen to extend to graphical models that are *a priori* suitable for populations of agents such as nanites.

Logics and Tools

Allied to algebraic calculi are new forms of mathematical logic, especially modal logics, specially designed to specify the properties that an interactive system should satisfy. Well-known example are dynamic logic [Pra], temporal logic [Pnu], the temporal logic of actions [LeL] and the mu calculus [Koz]. These logics often have a close link with algebraic calculi; an algebraic term denotes (part of) a system. while a logical formula says (in part) how it should behave. This underlies a successfully applied incremental methodology for system analysis; one verifies more and more properties of more and more parts (even the whole) of a system. Such verification is aided by software tools: model-checkers that can automatically verify properties of fairly complex finite-state systems [CES]; and semi-automated tools that can perform verifications with human guidance [CPS].

4 Safety and dependability

Nanites can disassemble, as well as assemble, structures. This has led to the notion of the so-called "grey goo" problem: nightmare visions of hordes of rogue nanites disassembling the wrong things, disassembling people, or even disassembling the entire planet. It is potentially the ultimate terrorist weapon.

Even if nanites are not deliberately engineered to be destructive, such objects will "naturally" appear in any replicating swarm of nanites. We are dealing with such vast numbers of nanites that some will spontaneously "mutate". Given the three features of reproduction, variation, and selection, some form of evolution will inevitably occur, leading to populations of "adjacent possible" undesigned nanites. Computer science, allied with biology, is crucial to the task of investigating and understanding these artificial evolutionary processes, and the defences we can design against them.

Dependability – the quality of a system that justifies its use even in critical conditions – is already a topic of extensive research in computer science. It involves mathematical analysis, as in the case of program verification and computer security; more widely, it involves making systems aware of, and able to report upon, their behaviour. It cannot exist without good modelling. The modelling of nanopopulations with dependability in mind, given their emergent properties and the inevitability of mutation, offers a huge challenge to CS.

Conclusion

Nanotech assemblers offer the promise of fantastic rewards. Some forms of nano-assemblers may well be exploitable and exploited in many ways without much CS input. Before we can achieve the full promise, however, there are many hard Computer Science problems to solve, concerning the design of emergent properties, the growth of physical artefacts, the programming and control of nanites, and defences against the "grey goo" and other safety critical scenarios.

References

[BeB] G. Berry, G. Boudol. The chemical abstract machine. *Theoretical Computer Science* 96, 217-248, 1992.

[BHR] S. D. Brookes, C. A. R. Hoare, A. W. Roscoe. A theory of communicating sequential processes. *Journal of the ACM*, 31, 560-699, 1984.

[CaG] L. Cardelli, A. Gordon. Mobile ambients. *Theoretical Computer Science*, 240, 177-213, 2000.

[CES] E. M. Clarke, E. A. Emerson, A. P. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on programming Languages and Systems*, 8(2),244-263, 1986.

[CPS] R. Cleaveland, J. Parrow, B. Steffen. The Concurrency Workbench: A Semantics Based Tool for the Verification of Concurrent Systems. *ACM ToPLaS*, 15, 36-72, 1993.

[D86] K. Eric. Drexler. *Engines of Creation: the coming era of nanotechnology*. Doubleday, 1986

[D92] K. Eric Drexler. *Nanosystems: molecular machinery, manufacturing and computation*. Wiley, 1992.

[Flo] R. W. Floyd. Assigning meanings to programs. *Mathematical Aspects of Computer Science, Proceedings of Symposia in Applied Mathematics* 19, American Mathematical Society, 19-32, 1967.

[Gor] M. J. C. Gordon. HOL: A proof generating system for higher-order logic. *VLSI Specification, Verification and Synthesis*, Kluwer 1987.

[Hoa] C. A. R. Hoare. An axiomatic basis for computer programming. *CACM,* 14(1), 39-45, 1971.

[Kau] Stuart A. Kauffman. *Investigations*. OUP, 2000

[Koz] D. Kozen. Results on the propositional mu-calculus. *Theoretical Computer Science*, 27, 333-354, 1983.

[LeL] L. Lamport. The Temporal Logic of Actions. *ACM Toplas*, 16(3), 872-923, 1994.

[Mil] R. Milner. *A calculus of communicating systems*. Lecture Notes in Computer Science 92, Springer, 1980.

[MPW] R.Milner, J. Parrow, D. Walker. A calculus of mobile processes. *Information and Computation*, 100(1),1-77, 1992.

[Pet] C.A. Petri. Kommunikation mi automaten. *PhD Thesis, Technical Report 2*, Institut fur Instrumentelle Mathemematik, Bonn, 1962.

[Pnu] A. Pnueli. The temporal logic of programs. *Proceedings of FOCS*, 46-77, IEEE, 1977.

[Pra] V.R. Pratt. Semantical considerations on Floyd-Hoare logic. *Proc. 17th Symposium on Foundations of Computer Science*, IEEE, 109-121, 1976.

[PrL] P. Prusinkiewicz, A. Lindenmayer. *The Algorithmic Beauty of Plants*. Springer, 1990.

[Rey] J.C. Reynolds. Towards a theory of type structure. *Proceedings of Paris Symposium on Programming*, Lecture Notes in Computer Science 16, 408-425, 1974.

[ScS] D. S. Scott, C. Strachey. Towards a mathematical semantics for computer languages. *Proceedings of Symposia on Computers and Automata, Microwave Research institute Symposia* 21, 19-46, 1971.

[Ste] Susan Stepney. Critical Critical Systems. In *Formal Aspects of Security, FASeC'02*. Lecture Notes in Computer Science, vol 2629. Springer, 2003.

[voN] J. von Neumann. *Theory of self-reproducing automata*. University of Illinois Press, 1966.