Unconventional Complexity Classes in Unconventional Computing
Extended Abstract

Antonio E. Porreca[0000−0003−1544−028X]

Aix-Marseille Université, CNRS, LIS, Marseille, France
antonio.porreca@lis-lab.fr

Abstract. Many unconventional computing models, including some that appear to be quite different from traditional ones such as Turing machines, happen to characterise either the complexity class **P** or **PSPACE** when working in deterministic polynomial time (and in the maximally parallel way, where this applies). We discuss variants of cellular automata and membrane systems that escape this dichotomy and characterise intermediate complexity classes, usually defined in terms of Turing machines with oracles, as well as some possible reasons why this happens.

1 Introduction

Unconventional computing models, particularly biologically inspired ones, appeared very early in the history of modern computer science, notably with cellular automata. Once new a model has been introduced, it is natural to analyse whether it is universal, in the sense that it can compute anything a Turing machine can; this happens quite often, since very little is actually needed in order to achieve universality (for instance, it usually suffices to simulate incrementation and conditional jump instructions [10]). These models are thus unconventional in terms of mechanism rather than in terms of computing power.

Nonetheless, a "conventional" computing power can be achieved with unconventional efficiency. This is where computational complexity questions become relevant. Some unconventional models, mostly sequential ones or parallel ones with a polynomial number of "processors", can simulate and be simulated by deterministic Turing machines with a polynomial-time overhead, and thus in particular they characterise the complexity class **P** when working in polynomial time. Emde Boas refers to those models as the *first machine class* [3]. This includes random-access machines with constant-time addition and subtraction [14] and, as detailed below, traditional cellular automata.

Other models, with less restrictions on parallelism or with more powerful elementary operations, characterise in polynomial time what deterministic Turing machines compute in polynomial *space*. This is called the *second machine class* [3] and includes parallel models such as tree-shaped hierarchies of processes generated by the `fork` system call under Unix (and running on an unbounded number of processors), as well as sequential ones such as random-access machines with

constant-time multiplication and division [2], and nondeterministic one such as alternating Turing machines [14].

Not all unconventional (or conventional, for that matter) computing models fall either into the first or the second machine class, but it happens often enough that one might be interested in investigating what features cause this behaviour, and what can be added, for instance, to a sequential model of the first class in order to solve more problems efficiently without always obtaining **PSPACE**[1]. One way to do that is to introduce randomisation [14, Chapter 11] or quantum computing features [19], but we will explicitly exclude these from the present discussion, both because they are outside the area of expertise of the author, and because they require an alternative definition of output which takes "wrong guesses" into account. Rather, let us focus on deterministic models and consider two examples, from the theory of cellular automata and from membrane computing.

2 Cellular Automata

One-dimensional cellular automata can simulate Turing machines with a polynomial-time slowdown, trivially by having a large enough set of states (or, equivalently, neighbourhood size) and storing the symbols, state, and tape head position of the Turing machine being simulated as states of the automaton[2] [17]. This is, however, a strictly sequential simulation carried out by a parallel model, and one might wonder whether cellular automata can be significantly more efficient when that parallelism is actually exploited. The answer is, however, readily seen to be negative: for any integer d, a d-dimensional cellular automata starting from a finite (non-quiescent) initial configuration of diameter $2r$ can always be simulated sequentially in polynomial time with respect to r. This happen because the volume of the smallest sphere containing the non-quiescent portion of the configuration is polynomial with respect to the radius, specifically $O(r^d)$, and the radius can only increase by one at each computation step of the automaton. Standard d-dimensional cellular automata are thus first class machines, despite their parallelism.

In order to solve harder problems we must then switch to cellular automata over non-hypercubic grids. The classic example from the literature is given by hyperbolic cellular automata [8]. Switching from Euclidean to hyperbolic geometry allows us to construct regular pentagonal grids, and the number of cells contained in a sphere of radius r is not polynomial anymore, but rather *exponential*. This allows many more cells to be active in parallel at any given time. More specifically, an infinite binary tree, whose branches represent communication channels, can be embedded in the pentagonal grid and exploited in order to solve **PSPACE**-complete problems in polynomial time. For instance, consider the quantified 3SAT problem for a formula of n variables: the idea is to explore

[1] Of course, here we work under the hypothesis that $\mathbf{P} \neq \mathbf{PSPACE}$ and that the intermediate complexity classes mentioned later are also distinct from both of them.

[2] With a more sophisticated reasoning, one can prove that the rule 110 automaton can also efficiently simulate Turing machines [13].

the 2^n possible truth assignments along 2^n distinct paths in the tree, evaluating the formula under each assignment separately, then propagating the results back towards the root; during this backpropagation phase, the truth values in two adjacent paths are combined by conjunction or disjunction, depending on the alternation of quantifiers in the formula, and the final result is obtained at the root. This is essentially a simulation of the computation tree of an alternating Turing machine. As a consequence, a hyperbolic computation space brings cellular automata to the second machine class.

An interesting variant of cellular automata, introduced by Modanese and Worsch [12], falls between **P** and **PSPACE** in polynomial time. These are *shrinking and expanding cellular automata*, where a cell can not only update its state based on its neighbourhood, but also delete itself (shrinking); furthermore, new cells can be created between two existing ones (expanding). While the full model once again characterises **PSPACE** by simulating the computation trees of alternating Turing machines [12], disallowing shrinking decreases the efficiency of the model. Essentially, in this model the information can only be propagated towards the leaf of the simulated computation tree, but since the distance between leaves become exponential in time, the information cannot be propagated back towards the root, and an unanimity acceptance condition is used instead [11]. As a result, expanding cellular automata characterise in polynomial time the class of problems *truth-table reducible to* **NP** *problems*. This is conjecturally weaker than **PSPACE**, but large enough to include both **NP** and **coNP** and, as such, fits our intuition of *unconventional complexity class*.

3 Membrane Computing

Membrane systems [15], also called P systems, are models inspired by the internal structure and functioning of biological cells. In their basic models, they consist of a tree-shaped hierarchy of nested membranes containing a *multiset* of molecules. The systems evolves inside each membrane by applying multiset-rewriting rules inspired by biochemical reactions; furthermore, membranes are selectively permeable, and can send molecules to, or receive them from adjacent regions. Finally, membranes can *divide* by fission, and their content is duplicated (with the exception of the triggering molecules, which differentiates the resulting membranes). These computation rules are applied, by default, in the *maximally parallel way*: if a molecule or membrane *can* be subject to at least one rule, then it *must* do so[3].

When no membrane division occurs, but only chemical reactions, the resulting systems belong to first machine class [20,1]. However, when membranes can divide

[3] Strictly speaking, if multiple rules are applicable, then one is chosen nondeterministically, but a *confluence* condition is usually imposed when solving decision problems (all computations must give the same result in the end). In all interesting cases known to the author, the systems can actually be made deterministic, although a general proof of this statement is still missing.

at arbitrary nesting depths, thus duplicating whole nested membrane substructures, exponential-size tree-shaped structures can be created in polynomial time, which can be exploited in order to simulate alternating Turing machines and prove, once again, an exact characterisation of **PSPACE** in polynomial time [18].

By restricting communication in only one direction (from the inside to the outside, i.e., membranes can send out molecules but never absorb them), similarly to the restriction on expanding cellular automata, the efficiency of membrane systems decreases and we obtain a characterisation of $\mathbf{P}^{\mathbf{NP}}$ in polynomial time [6]. This is the class of problems solved in polynomial time by deterministic Turing machines with access to an oracle for an **NP** problem; this can also be viewed as the class of problems Cook-reducible to **NP**, which highlights the similarity with expanding cellular automata even more. Another unconventional complexity class!

A different constraint we can impose is not related to the direction of communication but rather on the depth of the tree we can build. If communication is bidirectional, but the membrane systems are required to be *shallow* (only one level of nested membranes) or, more generally, if membranes can only divide if they do not contain further membranes, then we obtain a characterisation [4] of the complexity class $\mathbf{P}^{\#\mathbf{P}}$; this is the class of problems solvable in polynomial time with access to an oracle for a counting problem [14], or equivalently those Cook-reducible to **#P**.

4 Conclusions

The examples we described highlight how geometric (Euclidean vs hyperbolic space) and communication (monodirectional vs bidirectional) constraints can affect the efficiency of parallel unconventional computing models. Can these consideration be abstracted and formalised in a more general framework, or at least generalised to other communication topologies or geometries and for other computation models?

Another interesting aspect of the results on expanding cellular automata and membrane systems is that they provide a characterisation of complexity classes such as $\mathbf{P}^{\mathbf{NP}}$ and $\mathbf{P}^{\#\mathbf{P}}$ in terms of a *concrete, deterministic* model of computation; by this, we do not necessarily mean a *realistic* model (whenever exponentially many processors are needed, the shape of the approximately Euclidean space we live in is always a bottleneck [5]) but a deterministic model whose instantaneous configurations can be described in full detail, quite unlike the original definition of these complexity classes in terms of black-box oracles. Can other "abstract" complexity classes like these be captured by more concrete computing models?

Acknowledgements

This work was partially supported by the EU project MSCA-SE-101131549 "ACANCOS".

References

1. Alhazov, A., Leporati, A., Mauri, G., Porreca, A.E., Zandron, C.: Space complexity equivalence of P systems with active membranes and Turing machines. Theoretical Computer Science **529**, 69–81 (2014), https://doi.org/10.1016/j.tcs.2013.11.015
2. Bertoni, A., Mauri, G., Sabadini, N.: A characterization of the class of functions computable in polynomial time on random access machines. In: STOC '81 Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing. pp. 168–176 (1981), https://doi.org/10.1145/800076.802470
3. van Emde Boas, P.: Machine models and simulations. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science. Volume A: Algorithms and Complexity, pp. 1–66. Elsevier (1990)
4. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Simulating elementary active membranes, with an application to the P conjecture. In: Gheorghe, M., Rozenberg, G., Sosík, P., Zandron, C. (eds.) Membrane Computing, 15th International Conference, CMC 2014. Lecture Notes in Computer Science, vol. 8961, pp. 284–299. Springer (2014), https://doi.org/10.1007/978-3-319-14370-5_18
5. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Tissue P systems in the Euclidean space. In: Gheorghe, M., Petre, I., Pérez-Jiménez, M.J., Rozenberg, G., Salomaa, A. (eds.) Multidisciplinary Creativity: Homage to Gheorghe Păun on His 65th Birthday, pp. 118–128. Editura Spandugino (2015)
6. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Monodirectional P systems. Natural Computing **15**(4), 551–564 (2016), https://doi.org/10.1007/s11047-016-9565-2
7. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: Characterising the complexity of tissue P systems with fission rules. Journal of Computer and System Sciences **90**, 115–128 (2017), https://doi.org/10.1016/j.jcss.2017.06.008
8. Margenstern, M.: Cellular automata in hyperbolic spaces. In: Adamatzky, A. (ed.) Advances in Unconventional Computing, pp. 343–389. Springer (2017), https://doi.org/10.1007/978-3-319-33924-5_14
9. Martín-Vide, C., Păun, G., Pazos, J., Rodríguez-Patón, A.: Tissue P systems. Theoretical Computer Science **296**(2), 295–326 (2003), https://doi.org/10.1016/S0304-3975(02)00659-X
10. Minsky, M.: Computation: Finite and Infinite Machines. Prentice-Hall (1967)
11. Modanese, A.: Complexity-theoretic aspects of expanding cellular automata. Natural Computing **21**, 53–65 (2022), https://doi.org/10.1007/s11047-020-09814-2
12. Modanese, A., Worsch, T.: Shrinking and expanding cellular automata. In: Cook, M., Neary, T. (eds.) Cellular Automata and Discrete Complex Systems, 22nd IFIP WG 1.5 International Workshop, AUTOMATA 2016. Lecture Notes in Computer Science, vol. 9664, pp. 159–169. Springer (2016), https://doi.org/10.1007/978-3-319-39300-1_13
13. Neary, T., Woods, D.: P-completeness of cellular automaton rule 110. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006. pp. 132–143. Springer (2006), https://doi.org/10.1007/11786986_13
14. Papadimitriou, C.H.: Computational Complexity. Addison-Wesley (1993)
15. Păun, G.: Computing with membranes. Journal of Computer and System Sciences **61**(1), 108–143 (2000), https://doi.org/10.1006/jcss.1999.1693

16. Păun, G.: P systems with active membranes: Attacking NP-complete problems. Journal of Automata, Languages and Combinatorics **6**(1), 75–90 (2001)
17. Smith, A.R.: Real-time language recognition by one-dimensional cellular automata. Journal of Computer and System Sciences **6**(3), 233–253 (1972), https://doi.org/10.1016/S0022-0000(72)80004-7
18. Sosík, P., Rodríguez-Patón, A.: Membrane computing and complexity theory: A characterization of PSPACE. Journal of Computer and System Sciences **73**(1), 137–152 (2007), https://doi.org/10.1016/j.jcss.2006.10.001
19. Watrous, J.: Quantum computational complexity. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 7174–7201. Springer (2009), https://doi.org/10.1007/978-0-387-30440-3_428
20. Zandron, C., Ferretti, C., Mauri, G.: Solving NP-complete problems using P systems with active membranes. In: Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) Unconventional Models of Computation, UMC'2K, Proceedings of the Second International Conference, pp. 289–301. Springer (2001), https://doi.org/10.1007/978-1-4471-0313-4_21