Computability and Non-computability Issues in Amorphous Computing

Jiří Wiedermann

Computability and Non-Computability Issues in Amorphous Computing*

Jiří Wiedermann

Institute of Computer Science, Academy of Sciences of the Czech Republic,
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
jiri.wiedermann@cs.cas.cz

Abstract. Amorphous computing systems consist of a huge set of tiny simple stationary or mobile processors whose computational, communication and sensory part is reduced to an absolute minimum. In an airborne medium the processors communicate via a short-range radio while in a waterborne medium via molecular communication. In some cases the computational part of the processors can be simplified down to finite state automata or even combinatorial circuits and the system as a whole can still possess universal computational power with a high probability. We will argue that the amorphous systems belong among the simplest (non-uniform) universal computational devices. On the other hand, it is questionable as to what extent the standard universal models of computation can faithfully capture the behavior of amorphous computing systems whose functionality also depends on the non-computational and/or unpredictable operations of certain parts of the entire system.

1 Introduction

The notion of amorphous computing systems, i.e., of computational systems lacking any concrete "architecture", has emerged by the end of the 1990's. Initially, the development of such systems started as an engineering enterprise motivated by technological advancement in the field of micro-electro-mechanical systems, wireless communications and digital electronics (cf. [1], [2], [4], [5], [6], [7], [12], [13], [14]). Technological progress enabled integration of sensing, data processing and wireless communication capabilities into a single processor. In these systems the miniaturization has been pushed to its limits resulting, presumably, into processors of almost molecular size with the respective communication and computing facilities adequately (and thus, severely) restricted. These limitations alone, and the fact that systems consisting of huge numbers of processors are considered, have jointly called for the change of the basic computational and communication paradigms of distributed computing systems. These new paradigms also seem to have a potential to challenge certain computability issues related to our understanding of computing.

Nowadays we see amorphous computing systems in many forms (cf. [19]). Amorphous computing systems typically consist of a huge set of tiny simple processors

* This work was partially supported by RVO 67985807 and the GA ČR grant No. P202/10/1333

equipped with small memory, random number generator, simple wireless communication means, sensors and an energy source. The processors are randomly distributed over a closed area in which they can move, or are static. In the former case, the processors move by their own locomotion means, or by external forces, like Brownian motion, wind or stream. In an airborne medium the processors communicate via radio while in a waterborne medium via molecular communication. Moreover, in order to operate as envisaged some amorphous computing systems may exploit non-computable features, i.e., operations that cannot be realized computationally, such as self-replication or disintegration.

In this note we will focus our attention on specific instances of amorphous computing systems by which we will illustrate some remarkable aspects of amorphous computing systems.

First, we will be interested in their universality, i.e., in their ability to simulate arbitrary computations (of a Turing machine, say). Namely, for some amorphous computing systems this ability is by far not obvious due to the drastic restrictions imposed on computational and communication parts of the system's processors.

Strictly speaking, many (envisaged) applications of amorphous systems do not require universality. Single-purpose systems enabling, e.g., centralized data collection, a geographic area monitoring, intra-body multi-modal health monitoring, or drug delivery to certain body locations, often do. Nevertheless, universality qualifies these systems among the programmable systems which are, in principle, capable to perform an arbitrary algorithmic or even robotic task.

The second intriguing feature which we will be interested in is the problem of the reverse simulation of amorphous computing systems by standard models of universal computations. Namely, by their very nature, some amorphous computing systems are in fact described as physical (rather than purely computational) systems whose operation also depends on abilities of their processors and those of their environment that are of non-computational nature. This appears to be a serious obstacle for their faithful simulation on standard computational devices.

In the sequel, in Section 2 we will briefly describe two relatively advanced and unusual amorphous computing systems: an airborne, so-called flying amorphous computing system, and a waterborne system, so-called nanomachines. Here we will only provide a high-level description of the respective machines and of their computational and non-computational mechanisms in order to give the reader the main ideas of their functioning. In a more detail these systems have been introduced in earlier writings of the present author (cf. [11], [18]). Based on the previous descriptions, we will discuss the related universality issues in Section 3. In Section 4 we will concentrate on the problem of a reverse simulation of the previously described amorphous systems on a universal computational model. Conclusions are in Section 5.

2 Universality in Amorphous Computing Systems

2.1 Flying Amorphous Computer

A flying amorphous computer consists of a set of asynchronous processors. Each processor posses a clock running with the same speed as all the other processors; however,

the "ticking" of all clocks is not synchronized. Each processor is modeled by a "miniature" RAM with a finite number of registers capable of storing integers up to size N, with N denoting the number of nodes of the underlying amorphous system. Each processor is equipped by a random number generator and a single-channel radio device of a limited communication range. Initially, the processors have no unique identifier.

A severe restriction is imposed on the communication abilities of processors. A processor P_1 could receive a message sent by processor P_2 if and only if the following conditions hold true: (i) the processors are in the communication range of each other, (ii) P_1 is in a listening mode, and (iii) P_2 is in a broadcast mode, and it is the only processor within the communication radius of P_1 broadcasting at that time.

There is no mechanism making it possible to distinguish the case of no broadcast from that of broadcast collision. These restrictions concerning the radio communication are among the weakest ones that one can expect to be fulfilled by any simple radio communication device. The expected benefit from such restrictions is a simple engineering design of processors.

Note that the communication among the processors is complicated by the fact that the processors work asynchronously, have no identifiers, communication is one-way only, and there is no broadcast collision detection mechanism. As long as the processors remain anonymous (i.e., have no identifiers) a broadcasting processor has no means to learn that its message has been received by some processor. Under such circumstance, the randomized protocol designed in [10] enabling a reliable delivery of a message among processors within the communication range of each other works with a high probability, as follows.

The key idea is that the processors should broadcast a message sporadically in order to prevent message delivery (i.e., broadcast) conflicts, and repeatedly in order to maximize the likelihood of a successful delivery. The analysis of such a protocol reveals that the probability of sending should depend inversely on the expected number of a node's neighbors and should be repeated more times to handle the case of more processors in a node's neighborhood (cf. [17]).

Now, let us assume that cN of such processors, $c > 1$, fly around randomly in a confined convex volume. They form a dynamic network with a variable topology. The nodes of the network are created by processors with wireless communication links emerging asynchronously among the processors that find themselves within the communication radius of each other and fulfill the restriction for a successful one-way communication mentioned before. Our goal is to program the processors in such a way that they all together can simulate a RAM with N registers. Doing so, each RAM register will be represented in one processor of the flying computer.

The main problem is to keep the system operating under steadily changing topology of the communication network where new communication paths emerge, while the previous ones vanish. Some nodes may even become temporarily inaccessible since they may not find themselves within the communication range of other nodes. The latter problem can be solved under the assumption that no node in the network remains for ever isolated.

Thanks to this assumption, once processors do possess unique addresses, a message sent to a node with a given address would in a finite time reach this node and this node could send an acknowledgment that in a finite time will reach the sender.

The schema of the simulation is as follows. There is one specific node, a so-called leader. First, the leader invites all nodes to generate a random number within the range $[1..cN]$. Such an invitation is realized by "flooding" the net by an appropriate signal reaching all nodes with a high probability using the previously described protocol. Doing so, we cannot make use of the acknowledgments (since addresses are not yet available) and therefore a sufficient time must be allowed for the signal to spread over the entire volume with a high probability. Once the addresses are generated, the acknowledgment mechanism is used in all subsequent computations. Next, the duplicates are eliminated by a randomized algorithm described in [11] and the addresses are transformed into the range $[1..N]$. Now the simulation itself can start. It is a relatively straightforward procedure in which the next step is initiated by the leader only after the sender (i.e., the leader) obtains an acknowledgment from the receiver.

Although the whole system can correctly simulate a RAM (with a bounded memory size) with arbitrary high probability, the simulation time cannot be bounded by any function. However, if the address assignment process is successful (and this can be guaranteed with an arbitrary large probability), the simulation terminates within a finite time and always delivers the correct result. This computer has been described in full detail in [9] and later it was presented in [11].

2.2 Nanomachines

Recent unmatched improvements in nanotechnologies have enabled serious consideration of nano-scale machines whose size is of order 10^{-6} mm. Their prospective fabrication will make use of molecular self-assembly or of modifications of real bacteria via genetical engineering. To get an idea about the dimensions of objects we are considering, the size of a real bacteria is of the order of a few micrometers (i.e., of thousandths of millimeter, 10^{-6} m) while the size of a molecule is of the order of nanometers (i.e., 10^{-9} m). Thus, a nanomachine is about 1000 times bigger than a molecule and its surface and volume is still larger by a few orders of magnitude.

Next we will briefly describe so-called *self-reproducing mobile embodied automata* (nanomachines for short) whose information exchange mechanism is based on molecular communication.

Each nanomachine consists of two main components: there is its *embodiment* — the body, and its *computational part*.

The embodiment of any nanomachine consists of a set of receptors and emitters (pores), internal sensors, a set of timers, a self-reproducing mechanism, random bit generator and possibly of other devices depending on the type of embodiment (e.g., locomotive organs in the form of flagella, cilia, etc.).

Each receptor is specialized for detection of a specific type of molecules. These molecules find themselves in the environment surrounding the machine. Both the machines and the molecules move by convection (diffusion and advection). Moreover, the nanomachines can also move by their own means. For each type of molecules each

nanomachine has at its disposal several tens of receptors; their exact number is irrelevant. A molecule gets recognized only in the case when it enters into contact with the respective receptor.

Timers are internal mechanisms ("organs") without any external input. Each timer is preset for a fixed time. Each timer returns either 0 or 1. A timer can be reset to 0 by the machine's finite state control. Upon expiration of time for which the timer has been initially set the timer returns 1. Values to which the timers are preset depend on the type of a timer as well as on the properties of the environment (especially on its volume, but also on the properties of some molecules detected by the sensors — e.g., on the degradation time of the molecules). Timers of the same type are the same in all nanomachines.

The self-reproducing mechanism is a non-computational mechanism which is triggered by automaton entering a special reproducing state. In such a case, the nanomachine splits into two identical copies of itself, with their finite controls entering the initial state.

The random bit generator is an "organ" that upon each activation returns either 0 or 1 with the same probability.

The computational part of each nanomachine is created by a finite-state (Mealy) automaton whose actions are controlled by a transition function. In a single move each automaton reads its inputs obtained from its receptors and from other sensors or organs. Depending on these inputs and on its current state, the automaton issues instructions for the machine's organs concerning their next actions: releasing the molecules from the receptors, secreting signal molecules via the pores (output ports), resetting the timers, and instructing its locomotive organs. Last but not least, the control enters a new (possibly a reproduction) state. The use of timers and of a random number generator effectively turns the automata at hand into timed probabilistic automata.

Thus, the instructions for the machines are transmitted via elements from a finite set of molecular signals. Prior to sending a new signal, the environment must be cleared of the previous signal molecules. This is done by endowing the molecules with a certain self-destruction ability — after a certain time they spontaneously disintegrate into components that are not interpreted as any signals. These components are continuously absorbed by nanomachines and re-cycled inside their bodies in order to produce other molecular structures.

During their operation the self-reproducing mobile nanomachines communicate via so-called *quorum sensing,* i.e., by making collective decisions based on the density of nanomachine population. This density is inferred from the concentration of signal molecules emitted by the machines within a confined space.

In a given volume the machines multiply and emit the signal molecules until their maximal concentration has been reached. Then, they make a collective decision in which they simulate one step of a counter automaton. The resulting amorphous system was shown to be able to model a counter automaton [18], [15]. Thus, sequences of nanomachine populations of growing size obey a universal computing power.

Except of the timers and an organ serving as a random bit generator a further modification of the embodiment of the underlying automata may include a memory organ. Then, the task of memorizing the current state can be delegated to that organ. Conse-

quently, the computational mechanism of each nanomachine could be simplified down to combinatorial circuits of bounded depths (circuits from the complexity class AC_0).

The last mentioned model is of interest not only from a practical point of view, since it could lead to a simpler engineering of nanomachines, but also from the viewpoint of the theory of universal computing machines, as we will see in the sequel.

3 What is the simplest universal computational model?

For many computer scientists, a Turing machine, or a counter machine (also known as Minsky machine [8]) is considered as the simplest computational model possessing a universal computing power. In fact, there exists a plethora of universal computational devices and it is a matter of taste to select the simplest one from among those models. As a rule, they are represented by the devices with a fixed rigid architecture that enables them to enter configurations from a potentially infinite set of configurations: they are, in fact, infinite automata. None finite device (finite even in the physical meaning of this word) can subsequently enter an unbounded number of different configurations. Inevitably, the size of the device must grow with the number of reachable configurations. This is also the case of the cellular automata possessing a universal computing power. Note that for each input the corresponding cellular automaton is finite; however, the number of its elements grows with the input size.

The next (in)appreciable property of the known universal computational systems is their "non-homogeneity" — they cannot be disassembled into smaller, in some sense elementary identical computational parts that could be reassembled in an arbitrary manner so as to give rise to a universal computing system. A partial departure from this rule is given by cellular automata that almost satisfy our requirement of homogeneity. Clearly, a cellular automaton can be disassembled into individual finite automata. It is obvious that finite automata are simpler computational devices than, e.g., Turing machines — they only accept regular languages and, therefore, are not universal. Nevertheless, upon a re-assemblage into a cellular automaton one must respect the original topology of the underlying network although the individual automata need not be returned to their original positions — they can be interchanged at wish. Then a universal computing power "emerges" again. Note that in the latter case the universal computing power will not emerge from among any (multi)set of arbitrarily connected finite automata — the condition for that to happen is that the the communication links among the automata must follow a certain regular pattern.

The last consideration brings us to the following reformulation of the question from the title of this section: does there exist a simple computational device whose multisets possess universal computational power even when there is no fixed topology of communication links among their individual elements?

The answer to this question had been prepared in the previous two subsections. Both flying amorphous computers and the nanomachines are the candidates for such devices. This answer must further be stated more precisely. First, in order to ensure that computations of arbitrary space complexity could be realized we must always speak of sequences (or populations) of growing size of such systems. (The corresponding

devices are called non-uniform computational devices.) Second, we can only speak of simulations achieving their goal with a high probability.

If we had to make a choice between the two amorphous computing systems standing as candidates for the position of the simplest universal computing device the priority should probably be given to the population of nanomachines controlled by circuits. This is because their activity is governed by the simplest computing devices (viz. circuits). However, there is a price we have to pay for this simplicity. This is the fact that in addition to the purely computational part there is a non-computational mechanism comprising the "body" of the corresponding units of the system. In both cases, this body mainly consists of a communication mechanism, and in the case of nanomachines, the body also contains other, more complicated non-computational components, such as a self-reproducing mechanism, locomotive organs, etc. Moreover, in the case of nanomachines, a "collaboration" of the signal molecules — their disintegration in due time, was necessary. The operation of the system as a whole has been achieved by cooperation and orchestration of activities of all participating components controlled by computational parts of the units. From this point of view, cellular automata can bee seen as highly idealized models of amorphous systems considered in this paper in which it has been abstracted from the embodiment and communication mechanisms.

We conclude that amorphous computing systems considered above belong among the simplest (non-uniform) universal computing devices since their functionality is fully defined by the functionality of any of its parts, and there is no need to describe the "architecture" of the system as a whole.

4 The Problematic Simulation of Amorphous Systems by Turing Machines

When speaking about the universal computing power of amorphous computing systems in the previous sections we indicated how such systems can simulate devices which are already known to possess such a power. For the purpose of their reverse simulation, thanks to their embodiment and many non-computational features, amorphous computing systems should better be regarded as physical, rather than abstract mathematical systems. Thus, in this case, simulation should bridge the gap between a physical system and an abstract mathematical system. The crux of the problem is that behavior of the underlying physical system cannot be formally described down to the smallest detail. This is a source of difficulties when considering the reverse simulation of amorphous computing systems on universal models of computation.

What we have to do in this case is to duplicate, or imitate the functioning of amorphous computing systems on some formal model of computation in such a way that the behavior of the latter system closely mimics the observed behavior of the former system. Unfortunately, this appears to be practically impossible due to the unpredictable behavior of important elements of amorphous computing systems. For instance, in the case of flying amorphous computer, the trajectories of processors are continuous and random as well as the asynchronous communication activities of processors. All these activities run in parallel and concurrently. Moreover, e.g., in the case of nanomachines, the degree of parallelism increases with time (in the first phase of rising up the popula-

tion of nanomachines). There are additional parallel processes running at each nanomachine (such as signal molecules sensing/emitting) and also those corresponding to the interactions of molecules in the environment. The disintegration processes of signal molecules must also be taken into account. It is difficult to imagine how emulation of such processes could run on a sequential computer (or any other computer with a bounded parallelism) keeping track of a potentially unbounded number of spatially and temporally related physical variables. Discretization of continuous processes underlying the operation of amorphous computing systems (e.g., think of the movement of the processors) within a discrete computational model, asynchronicity and a potentially unbounded parallelism may thus introduce unsurmountable timing problems for simulation of such processes.

These facts challenge the popular belief that any physical computational device can be emulated within any other, and especially, within any universal model of computation (cf. [3] and the references therein).

Amorphous computing systems also offer an interesting answer to the question ”what everything can compute?”. At the same time, they seem to present an example of computations that is captured neither by Turing machines nor by any of their parallelized or other known variants. Obviously, any answer to the question ”what is computation?” should also cover computations of amorphous computing systems.

5 Conclusions

Amorphous computing systems confront us with an interesting dichotomy. They harness non-computational mechanisms for the purpose of computing, but they also exploit computing in order to control these non-computational mechanisms. Thus, amorphous computing systems present a class of computing systems in which physical aspects, manifested through their embodiment, play an important role. Amorphous computing systems can be proved to be computationally universal with a high probability. The respective proofs can come through thanks to focusing to certain computational aspects of their functionality and postulating the expected outcomes of non-computational operations that also contribute to the mechanism of computation. Thus, when reasoning about computational universality of amorphous systems we reason, in fact, about a more abstract, simplified, often probabilistic model of an amorphous system.

The situation changes when we want to capture the behavior of a “real” amorphous computing system (as opposed to that of its model) on a universal computer. This might be the case when we want, e.g., to tune some physical parameters of an amorphous system in order to achieve its better practical performance. In such a case, we have to simulate the physical system as it is, i.e., inclusively of its non-computational aspects which are important for keeping the entire system performing its computational task. The non-computational aspects are determined by complex physical and in some cases, also chemical interactions among the basic elements of amorphous systems which cannot be described as computational processes.

An additional problem in simulating (or more precisely: in emulating) an amorphous computing system stems from the fact that such systems, in dependence on the input size, are capable to perform a potentially unbounded number of parallel opera-

tions in constant time. This cannot be done by any uniform physical model of a known parallel universal computer with a constant number of processors. Perhaps the self-reproduction mobile embodied automata present the first step towards a truly universal computational model.

References

1. H. Abelson, et al. Amorphous Computing. MIT Artificial Intelligence Laboratory Memo No. 1665, Aug. 1999
2. H. Abelson, D. Allen, D. Coore, Ch. Hanson, G. Homsy, T. F. Knight, Jr., R. Nagpal, E. Rauch, G. J. Sussman, R. Weiss. Amorphous Computing. Communications of the ACM, Volume 43, No. 5, pp. 74–82, May 2000
3. Nagy, N., Akl, S.G.: Time inderterminacy, non-universality in computation, and the demise of the Church-Turing thesis, School of Computing, Queen's University, Kingston, Ontario, Technical Report No. 2011-580, August 19, 2011, 27 p.
4. D. K. Arvind, K. J. Wong: Speckled Computing A Disruptive Technology for Network Information Appliances. Proc. IEEE International Symposium on Consumer Electronics (ISCE'04), 2004, pp. 219-223
5. D. Coore: Introduction to Amorphous Computing. Unconventional Programming Paradigms: International Workshop 2004, LNCS Volume 3566, pp. 99–109, Aug. 2005
6. J. M. Kahn, R. H. Katz, K. S. J. Pister. Next century challenges: mobile networking for "Smart Dust". In: Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking, MobiCom '99, ACM, pp. 271–278, Aug. 1999
7. J. M. Kahn, R. H. Katz, K. S. J. Pister. Emerging Challenges: Mobile Networking for Smart Dust. Journal of Communications and Networks, Volume 2, pp. 188–196, 2000
8. M. Minsky, *Computation: Finite and Infinite Machines,* Prentice-Hall, 1967
9. L. Petrů: Universality in Amorphous Computing. PhD Disseration Thesis. Dept. of Math. and Physics, Charles University, Prague, 2009
10. L. Petrů, J. Wiedermann: A Model of an Amorphous Computer and Its Communication Protocol. In: Proc SOFSEM 2007: Theory and Practice of Computer Science. LNCS Volume 4362, Springer, pp. 446–455, July 2007
11. L. Petrů, J. Wiedermann: A Universal Flying Amorphous Computer. In: Proc. Unconventional Computation, 10th International Conference, UC'2011, LNCS, Vol. 6714, 2011, pp. 189-200
12. M. J. Sailor, J. R. Link: Smart dust: nanostructured devices in a grain of sand, Chemical Communications, Vol. 11, p. 1375, 2005
13. S. C. Shah, F. H. Chandio, M. Park: Speckled Computing: Evolution and Challenges. Proc. IEEE International Conference on Future Networks, 2009, pp. 181-185
14. B. Warneke, M. Last, B. Liebowitz, K. S. J. Pister: Smart Dust: communicating with a cubic-millimeter computer. Computer, Volume: 34, Issue: 1, pp. 44–51, Jan. 2001
15. J. Wiedermann, L. Petrů: Computability in Amorphous Structures. In: Proc. CiE 2007, Computation and Logic in the Real World. LNCS Volume 4497, Springer, pp. 781–790, July 2007
16. J. Wiedermann, L. Petrů: Communicating Mobile Nano-Machines and Their Computational Power. In: Third International ICST Conference, NanoNet 2008, Boston, MA, USA, September 14-16, 2008, Revised Selected Papers, LNICST Vol. 3, Part 2, Springer, pp. 123-130, 2009.
17. J. Wiedermann, L. Petrů: On the Universal Computing Power of Amorphous Computing Systems. Theory of Computing Systems 46:4 (2009), 995-1010, www.springerlink.com/content/k2x6266k78274m05/fulltext.pdf

18. J. Wiedermann: Nanomachine Computing by Quorum Sensing. In: Computation, Cooperation, and Life. Essays Dedicated to Gheorghe Păun on the Occassion of His 60th Birthday. J. Kelemen, A. Kelemenová, eds. LNCS Vol. 6610, Springer, pp. 203-215
19. J. Wiedermann: The Many Forms of Amorphous Computational Systems. In: H. Zenil (Editor): A Computable Universe: Understanding and Exploring Nature As Computation. World Scientific Publishing Company, 2012, to appear