Brief Announcement: Amoebot: A New Model for Programmable Matter

Zahra Derakhshandeh
Arizona State University
zderakhs@asu.edu

Shlomi Dolev
Ben-Gurion University
dolev@cs.bgu.ac.il

Robert Gmyr
University of Paderborn
gmyr@mail.upb.de

Andréa W. Richa
Arizona State University
aricha@asu.edu

Christian Scheideler
University of Paderborn
scheideler@mail.upb.de

Thim Strothmann
University of Paderborn
thim@mail.upb.de

ABSTRACT

The term *programmable matter* refers to matter which has the ability to change its physical properties (shape, density, moduli, conductivity, optical properties, etc.) in a programmable fashion, based upon user input or autonomous sensing. This has many applications like smart materials, autonomous monitoring and repair, and minimal invasive surgery, so there is a high relevance of this topic to industry and society in general. While programmable matter has just been science fiction more than two decades ago, a large amount of research activities can now be seen in this field in the recent years. Often programmable matter is envisioned, as a very large number of small locally interacting computational *particles*. We propose the Amoebot model, a new model which builds upon this vision of programmable matter. Inspired by the behavior of amoeba, the Amoebot model offers a versatile framework to model self-organizing particles and facilitates rigorous algorithmic research in the area of programmable matter.

Categories and Subject Descriptors

F.m [**Theory of Computation**]: Miscellaneous

Keywords

programmable matter; self-organization; nano-computing; mobile robots

1. INTRODUCTION

Recent advancements in microfabrication and cellular engineering foreshadow that in the next few decades it might be possible to assemble simple information processing units at almost no cost. Myriads of these small-scale units could be combined to powerful systems to solve intricate tasks. This vision of building cheap microscopic processing units is supported by the progress made in manufacturing microelectronic mechanical components, such that one can anticipate integrating logic circuits, microsensors, actuators, and communications devices on the same chip. These devices could be mixed with bulk materials, such as paints, gels, and concrete, depending on the tasks these devices should solve. Aside from the microelectronic way of construction there has also been intriguing progress in understanding the biochemical mechanisms of individual cells such as the mechanisms behind cell signaling and cell movement. Recent results have also demonstrated that, in principle, biological cells can be turned into finite automata or even pushdown automata. Therefore, one can imagine to tailor-make biological cells to operate as sensors and actuators, as programmable delivery devices, and as chemical factories for the assembly of nano-scale structures.

But fabrication is only part of the story. One can envision producing vast quantities of microscopic individual computing elements but research on how to use them is effectively still in its infancy. The opportunity to exploit these new technologies poses a broad conceptual challenge that was coined by Toffoli and Margolous as *programmable matter* [9]. Many research directions related to computer science, from robotics to sensor networks to distributed computing to molecular self-assembly, are converging towards similar problems and ideas in this general research field. In particular, there is a strong emphasis on the parallel and distributed nature of the problems. Yet, there is still no clear understanding which tasks can be solved by microfabricated computing elements (in the sense of computability) or which problems can be solved efficiently. In the area of programmable matter one usually assumes that there is a very large number of locally interacting computational particles that may form an arbitrary initial structure. The particles are possibly faulty, sensitive to the environment, and may produce various types of actions that range from changing their internal state to communicating with other particles, sensing the environment, moving to a different location, changing shape or color, or even replicating, which may then be used to change the physical properties, color, and shape of the matter at a global scale. The research done in the general area of programmable matter is rich and multifaceted, ranging from passive systems like DNA computing, population protocols, and slime molds to active systems like self-organizing networks, swarm robotics and modular robotic systems. Our work focuses on building and exploiting a formal model which is closely related to the idea of having lots of computationally limited *particles* that can

WOODSTOCK '97 El Paso, Texas USA

bond to neighboring particles.

Despite significant work in the areas of swarm robotics (e.g., [6, 7]), DNA computing (e.g., [5, 8, 12]), self-assembled molecular computing (e.g., [13]), modular self-reconfigurable robotic systems (e.g., [3, 4, 11]), and other related areas, either the proposed models are not applicable to our scenario or they lack a formal presentation that allows rigorous algorithmic research (for more references on related work please see [1]).

We propose *Amoebot*, a new amoeba-inspired model for *programmable matter*, which consists of computational particles representing finite automata that form a connected structure with the help of local bonds. The particles *cannot move through other particles or in open space*, and while they move they *cannot drag other particles* with them due to limitations on their energy and strength. So if they wish to form a particular pattern, then the only way to achieve that is through *individual movements of particles along the surface of a particle structure*, which could be done by releasing and engaging bonds to static particles in a similar way as some biological cells move (e.g. amoebas [2] and some skin cells [10]) until they have reached the desired pattern. This is continued until the desired goal pattern has been reached.

2. MODEL

Consider the *equilateral triangular graph*, see Figure 1. A *particle* occupies either a single node or a pair of adjacent nodes in this graph, and every node can be occupied by at most one particle. Two particles occupying adjacent nodes are defined to be *connected*.

Every particle has a *state* from a finite set Q. Connected particles can communicate via the edges connecting them in the following way. A particle p holds a *flag* from a finite alphabet Σ for each edge that leaves p (i.e., all edges incident in a node occupied by p except the edge between the occupied nodes if p occupies two nodes). A particle occupying the node on the other side of such an edge can read this flag. This communication process can be used in both directions over an edge. In order to allow a particle p to address the edges leaving it, the edges are labeled from the local perspective of p. This labeling starts with 0 at an edge leading to a node that is only adjacent to one of the nodes occupied by p and increases counter-clockwise around the particle. The restriction for label 0 will be used below to uniquely define which edge is labeled 0 when a particle moves.

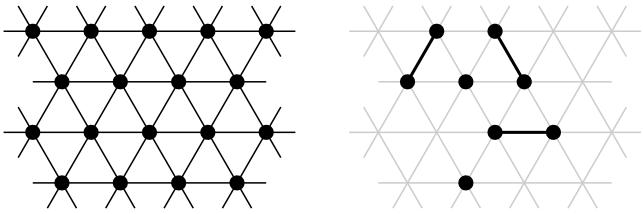

Figure 1: The left half of the figure depicts a section of the infinite equilateral triangular graph. Nodes are shown as black circles. The right half shows five particles on the graph. When depicting particles we draw the graph as a gray mesh without nodes. A particle occupying a single node is depicted as a black circle, and a particle occupying two nodes is depicted as two black circles connected by an edge.

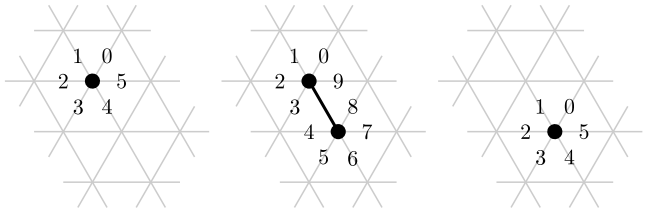

Figure 2: The three parts of the figure show a moving particle together with the labels seen by the particle. On the left, the particle occupies only a single node. The particle then expands in the direction of the edge labeled 4 resulting in the particle occupying two nodes as depicted in the middle. Since the expansion changes the number of edges leaving the particle, the edges have to be relabeled. The direction of the edge labeled 0 remains constant. Because of the restriction for the label 0 mentioned in the text, this uniquely defines the edge that will receive the label 0 after the expansion. Next the particle contracts out of one of the nodes it currently occupies towards the direction of the edge labeled 6 resulting in the particle occupying only a single node as depicted on the right. Again, the edges leaving the particle are relabeled.

Particles move through *expansion* and *contraction*: If a particle occupies one node, it can expand into an unoccupied adjacent node to occupy two nodes. If a particle occupies two nodes, it can contract out of one of these nodes to occupy only a single node. During these movements, the direction of the edge labeled 0 remains constant, even though the edge itself might change. Figure 2 shows an example of the movement of a particle. Besides executing expansions and contractions in isolation, we allow pairs of connected particles to combine these primitives to perform a coordinated movement: One particle can contract out of a certain node at the same time as another particle expands into that node. We call this movement a *handover*. The particles involved in a handover are defined to remain connected during its execution.

Computationally, particles resemble finite state machines. A particle acts according to a *transition function*

$$\delta : \; Q \times \Sigma^{10} \; \rightarrow \; \mathcal{P}(Q \times \Sigma^{10} \times M).$$

For a particle p the function takes the current state of p and the flags p can read via its leaving edges as arguments. Here, the i-th coordinate of the tuple Σ^{10} represents the flag read via the edge labeled i when numbering the coordinates of the tuple starting at 0. If for a label i there is no edge with that label or if the respective edge leads to a node that is not occupied, the coordinate of the tuple is defined to be ε. The value $\varepsilon \in \Sigma$ is reserved for this purpose and cannot be set as a flag by a particle. The transition function maps to a set of *turns*. A turn is a tuple specifying a combination of a new state to assume, new flags to set, and a movement to

execute. The set of movements is defined as

$$\begin{aligned} M = &\{\text{idle}\} \cup \\ &\{\text{expand}_i \mid i \in [0,9]\} \cup \\ &\{\text{contract}_i \mid i \in [0,9]\} \cup \\ &\{\text{handoverContract}_i \mid i \in [0,9]\}. \end{aligned}$$

The movement *idle* means that p does not move, and $expand_i$ and $contract_i$ are defined as mentioned above. The index i specifies the edge that defines the direction along which the movement should take place, as shown in the example in Figure 2. The movement $handoverContract_i$ specifies a contraction that can only be executed as part of a handover. Summarizing, a transition function specifies a set of turns a particle would like to perform based on the locally available information.

A system of particles progresses by executing atomic *actions*. An action is either the execution of an isolated turn for a single particle or the execution of a turn for each of two particles resulting in a handover between those particles. Note that if a movement is not executable, the respective action is not enabled: For example, a particle occupying two nodes cannot expand although it might specify this movement in a turn. As another example, a particle cannot expand into an occupied node except as part of a handover. Finally, an action consisting of an isolated turn involving the movement $handoverContract_i$ is never enabled as this movement can only be performed as part of a handover. The transition function is applied for each particle to determine the set of enabled actions in the system. From this set, a single action is arbitrarily chosen and executed. The process of evaluating the transition function and executing an action continues indefinitely.

3. DISCUSSION

The general Amoebot model as described in the previous section can take various specific forms depending on how systems of particles are initialized, what information particles keep track of in their state, and what information they share over their leaving edges. For example, the model does not specify how the labeling of the edges leaving a particle is initially set up. Always assigning the label 0 to an edge pointing upwards for every particle results in a full-compass model, while assigning this label differently for each particle results in a no-compass model where particles only share their sense of orientation. As an example for the influence of communication, in the no-compass variant of the model two particles can compare the relative rotation of the direction of their respective edges with label 0 by communicating, but this is by no means mandatory. As a final example, the particles can be set up to know whether they occupy one or two nodes, they can keep track of this information during the progression of the system, and they can communicate this information over their leaving edges, but again none of this is mandatory.

While the model allows for many variations, it also has fixed core features. Some of these features represent our idea on how programmable matter consisting of discrete particles would work: The particles have constant memory (in particular, they have no identifiers) and modest computational power. Particles act asynchronously and make their decision based on local information. Finally, particles can only move themselves and not other particles because of limited physical strength. However, the most striking feature of the Amoebot model, namely the use of expansion and contraction as locomotion primitives together with handovers as their combination, is motivated by the problems we want to investigate using the model: We want to study problems in which all particles in a system have to form a *single connected component* at all times. An example of why our type of locomotion is favorable in this context is that it allows arbitrarily long chains of (initially contracted) particles to move along a common trajectory without losing connectivity. All particles of such a chain simply expand and contract in alternation. The first particle in the chain determines the trajectory by deciding the direction of its expansion, the expansion of the remaining particles is always in direction of their predecessor in the chain. Furthermore, a contraction is always in the same direction in the grid as the previous expansion and all contractions are handover-contractions except the contractions of the last particle in the chain.

4. RESEARCH CHALLENGES

We would like to use the Amoebot model to investigate various problems in which the system of particles has to form a *single connected component* at all times. As an example, the class of *coating problems* might be considered. Here, the surface of an object is to be coated as uniformly as possible by the particles of a system. A second example is the class of *shape formation problems* in which a system has to arrange to form a specific shape. Finally, in *bridging problems* particles have to bridge gaps in given structures. We see the coating problems as an algorithmic primitive for solving other problems. For example, the formation of a shape can be achieved by creating an initially small instance of that shape which is then iteratively coated to form increasingly large instances until the number of particles in the system is exhausted.

Also, variants of the model might be of interest. Firstly, in a physical realization particles may become faulty and a system may not be well-initialized. For a system to handle such occurrences it may be necessary to allow particles to detect other faulty particles and to cope with them. Such a model would require self-stabilizing algorithms. Secondly, the model may be extended from two to three dimensions For more information, including preliminary results on coating problems and more related work please see [1].

5. REFERENCES

[1] amoebot.cs.upb.de.

[2] R. Ananthakrishnan and A. Ehrlicher. The forces behind cell movement. *International Journal of Biological Sciences*, 3(5):303–317, 2007.

[3] Z. Butler, S. Murata, and D. Rus. Distributed replication algorithms for self-reconfiguring modular robots. In *Distributed Autonomous Robotic Systems 5*, pages 37–48. Springer, 2002.

[4] Z. J. Butler, K. Kotay, D. Rus, and K. Tomita. Generic decentralized control for lattice-based self-reconfigurable robots. *International Journal of Robotics Research*, 23(9):919–937, 2004.

[5] K. C. Cheung, E. D. Demaine, J. R. Bachrach, and S. Griffith. Programmable assembly with universally foldable strings (moteins). *IEEE Transactions on Robotics*, 27(4):718–729, 2011.

[6] S. Kernbach, editor. *Handbook of Collective Robotics – Fundamentals and Challanges*. Pan Stanford Publishing, 2012.

[7] J. McLurkin. *Analysis and Implementation of Distributed Algorithms for Multi-Robot Systems*. PhD thesis, Massachusetts Institute of Technology, 2008.

[8] R. Nagpal, A. Kondacs, and C. Chang. Programming methodology for biologically-inspired self-assembling systems. Technical report, AAAI Spring Symposium on Computational Synthesis, 2003.

[9] T. Toffoli and N. Margolus. Programmable matter: concepts and realization. *Physica D: Nonlinear Phenomena*, 47(1):263–272, 1991.

[10] X. Trepat, M. R. Wasserman, T. E. Angelini, E. Millet, D. A. Weitz, J. P. Butler, and J. J. Fredberg. Physical forces during collective cell migration. *Nature physics*, 5(6):426–430, 2009.

[11] J. E. Walter, J. L. Welch, and N. M. Amato. Distributed reconfiguration of metamorphic robot chains. *Distributed Computing*, 17(2):171–189, 2004.

[12] E. Winfree, F. Liu, L. A. Wenzler, and N. C. Seeman. Design and self-assembly of two-dimensional dna crystals. *Nature*, 394(6693):539–544, 1998.

[13] D. Woods, H.-L. Chen, S. Goodfriend, N. Dabby, E. Winfree, and P. Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *ITCS*, pages 353–354, 2013.