Self-Organized Learning of Collective Behaviours in Agent-Based Simulations

Abbas Sarraf Shirazi, Sebastian von Mammen, Iman Yazdanbod, and Christian Jacob

Department of Computer Science
University of Calgary, Canada
{asarrafs,s.vonmammen,iyazdanb,cjacob}@ucalgary.ca

In this paper, we propose an approach to reduce the number of interactions among agents in a multi-agent simulation. Modeling agent interactions turns out to be computationally expensive, especially when arbitrary interactions are allowed. In order to diminish computational costs of simulating agent interactions, we propose a self-organized approach to abstracting recurrent interaction patterns among groups of agents.

Predictably interacting groups of agents are subsumed by higher-order agents that reproduce similar behaviours but at reduced computational costs. To this end, observer agents are immersed into the simulation space in order to monitor groups of agents and learn interaction patterns. Since the dynamics of the system changes over time, an abstraction might loose its validity and must therefore be removed again. This process is regulated by confidence values that are calculated and associated with individual abstractions. If a pattern exists for longer than anticipated, its confidence value is increased. The process of creating and removing abstractions is repeated during the course of a simulation in order to ensure an adequate adaptation to the system dynamics. Experimental results on a biological agent-based simulation show that our proposed abstraction method can successfully reduce the computational complexity during the simulation while maintaining the possibility of arbitrary interactions.

1 Introduction

In the context of this paper, we are particularly interested in complex systems, as they occur in biological processes and human physiology. In general, such systems are composed of hierarchically intertwined and interacting parts. Computational models can be implemented following the agent-based modelling approach (ABM) that provides each of these parts with the ability to change their own states and to interact with others. Agent-based computational models have gained great popularity as they can easily consider noise, spatial and temporal relationships and exhibit emergent processes [8, 3, 4].

The flexibility of agent-based models comes at a cost: Without restrictions, each agent could potentially interact with all the other ones. Merely identifying who interacts with whom then becomes a computationally expensive task, not even considering the actualisation of any interactions. Usually, numerous concurrent agent-based simulations are therefore limited to fixed neighbourhoods in discrete lattice spaces as implemented by cellular automata (CA). The ability of the models to continuously change the interaction topology among the agents, however, is crucial to trace, for instance, the dynamics of transportation effects [19] or developmental processes [15].

In this paper, we present a means to reduce the arising computational costs while preserving the flexibility of agent-based models. In particular, we show how groups of agents that exhibit behavioural patterns can be reduced to individual agents with (computationally) simplified interaction rules. In addition to the agents that are part of the actual simulation model, *observer agents* are immersed into the simulation space to monitor groups of agents, to learn their interaction patterns, and to temporarily replace them (at reduced computational costs). As the agents' interactions may change over time, learned behaviours might loose their validity. Therefore, confidence values determine the lifetime of the learned behavioural patterns. Continuous re-evaluation of these confidence values allows for a self-organized optimization process in which the substitutions, initiated by *observer agents*, are adaptively created and revoked.

The remainder of this paper is organized as follows. Section 2 reviews related work in the field of multi-scale and multi-agent modelling of biological systems. Section 3 presents the details of our proposed method. Section 4 reports on the experiments conducted on an agent-based blood coagulation simulation to study the performance of the proposed method. Finally, concluding remarks are presented in 5.

2 Related Work

Simulations can predict the behaviours of complex systems. The resulting phenomena emerge bottom-up from the interactions of the parts of a system, or its agents [4]. Although the natural sciences have uncovered numerous parts and processes that constitute natural systems, it is not possible to comprehensively represent or simulate them in accordance with the most recent findings. The

parts and processes across several dimensions of scale would need to be described by means of one modelling language and be integrated into one computational model. It has been suggested that hierarchical representations of agents could address these challenges, and that the associated levels of abstraction could even reduce the computational load of large-scale, bottom-up simulations. In this section, we briefly describe some of the related preceding works.

2.1 Bottom-up Models

Artificial chemistries [1] and computational developmental systems—such as L-systems [13], relational growth grammars [10], or swarm grammars [20] explicitly and often visually trace the emergence of high level structures based on simple constituents—these constituents may be represented as formal symbols or as entities in physics simulations. Even when looking at simplified physical artificial chemistries, only allowing for the most simple interactions between molecules, complex interaction patterns can emerge from the bottom-up.

Schuster coined the term hypercycle which denotes a system of chemicals and their reactions that nurture one another. Through the formation of such intertwined entities, hierarchies of increasing complexity emerge in nature [17]. Autocatalytic networks also pick up on this idea [9].

Rasmussen et al. designed a computational model based on artificial chemistries, in which structures are formed with an increase in structural complexity and with different functionalities, from monomers to polymers to micelles [14]. Although their experiments clearly suggest the formation of patterns at several levels of scale, Dorin and McCormack claim that such phenomena are not surprising given the model's simplicity. They further argue that it takes considerably more effort to determine the novelties at higher levels in the hierarchy [7].

2.2 Re-representing Emergent Patterns

Several conceptual approaches have been presented that address the issue of identifying emergent phenomena.

Servat et al. propose that an emergent phenomenon should be represented as an abstract agent ignoring unimportant details [18]. Although the authors suggest that externally observed changes could provide clues to introduce and configure high-order agents, they conclude that agents on a higher level must be predefined as their behaviours need to be known upfront.

Along the same lines, Chen et al. introduce an agent description formalism capable of capturing emergent patterns at different levels of abstraction [2, 3]. Although the existence of emergent patterns should be automatically and formally inferred, they also require that higher-level patterns are introduced as background knowledge into the system beforehand.

Lavelle et al. use the term immergence, or downward causation, to describe the effect of the higher level organization on entities at lower scales [11]. They

postulate that explicit functions must be defined to bridge between micro and macro levels.

2.3 Agent Group Substitution

In order to capture emergent phenomena, Dessalles and Phan foresaw a system in which detectors would identify emergent patterns and subsume the activity of the respective lower level objects [6].

von Mammen et al. introduced a conceptual model of self-organized middle-out abstraction (SOMO) in agent-based simulations [21]. In their proposed approach, observer agents monitor the interaction history of sets of agents, use motif discovery algorithms to detect recurrent patterns, and create hierarchies of high-level agents that subsume the lower interacting agencies. Although they do not exclude the possibility of a relationship between learned high-order patterns and emergent phenomena found in nature, their concept primarily targets an increase of efficiency by repeatedly substituting groups of agents by individual high-level instances that work at lower computational cost.

By means of a rudimentary prototype, the authors of this work have previously shown that high-level agent substitution indeed results in a reduction of computational costs [16]. In particular, they merged artificial neural network learning, an established inductive learning method, with an agent-based implementation of a signaling pathway. Clusters of biological substrates and their corresponding activation patterns were substituted by artificial neural network agents.

3 Self-Organized Learning of Collective Behaviours

In this section, we present an approach that maintains the capacity of letting arbitrary agent interactions unfold over time and adaptively reduces the computational costs that arise from unrestricted interaction possibilities.

In addition to the agents that constitute the simulation model, or *model agents*, we introduce *observer agents*, or *observers*, into the simulation space. The simulation framework treats both kinds of agents equally, i.e. each of these agents is considered for interactions at each simulation step, depending on the agents' behaviours and the simulation state.

Observers monitor subsets of *model agents* and look for patterns in their interactions. Once an *observer* successfully identifies an interaction pattern, it replaces the individual behaviours maintained by the *model agents* that led to this pattern by a group behaviour that it executes on behalf of the *model agents*. The execution of this group behaviour is based on the learned pattern instead of the *model agents*' situations, which results in a reduction of the computational costs.

Table 1: Interaction histories inside an *observer*

Interaction History of Executed Actions (IH_{Exec})			
Ag_0	**Activate**	$\mathcal{A}_1$	t_0
Ag_0	**Activate**	$\mathcal{A}_2$	t_1
.	.	.	.
.	.	.	.
.	.	.	.
Ag_2	**Activate**	$\mathcal{A}_2$	t_1
Ag_7	**Activate**	$\mathcal{A}_2$	t_2
Ag_{12}	**Activate**	$\mathcal{A}_2$	t_4

Interaction History of Computed but Unexecuted Actions (IH_{NExec})		
t_0	**Activate**	n_0
t_3	**Activate**	n_3
.	.	.
.	.	.
.	.	.
t_{15}	**Activate**	n_{15}
t_{23}	**Activate**	n_{23}
t_{32}	**Activate**	n_{32}

Since the substitution of the *model agents*' individual behaviours with the *observer*'s group behaviour diminishes the capacity of interactions in the simulation, the *observer* has to continuously validate it. For the validation step, the *observer* would simply check whether the deployment of the original individual behaviours would yield an outcome different from the predictions of the learned pattern. If the discrepancy between these two outcomes exceeds a given threshold, the *observer* omits its learned pattern and reactivates the original individual behaviours.

The success of the abstraction system depends on the configuration of the deployed *observer* agents. In the following paragraphs, we explain one way how the *observers* can replace the individual behaviours with a group behaviour and how they can validate, maintain, or abandon the learned patterns throughout the course of a simulation.

3.1 The Observer

Like any other agent, an *observer agent* can be defined as $Ag = (Sit, Act, Dat)$, a triple composed of a set Sit of situations, a set Act of actions, and a set Dat of internal data [5]. At any point in time, the agent decides to perform an action based on its situation and internal data. This decision is captured in a decision function $f_{ag} : Sit \times Dat \rightarrow Act$. Without the loss of generality, f_{Ag} can be represented as a set of situation-action pairs.

Observers are configured to log the interactions of *model agents* in their interaction histories: IH_{Exec} is used to log executed interactions, whereas IH_{NExec} logs the numbers of considered but not executed actions (Table 1). An IH_{Exec} entry may contain any information related to an observed interaction. For instance, an *observer* may store the *model agent* A that executed an action $act \in Act$ with time stamp t along with the set of interaction partners $\mathcal{A}$. In addition, the *observer* could also log situational information of the *model agent* and its interaction partners (Figure 1).

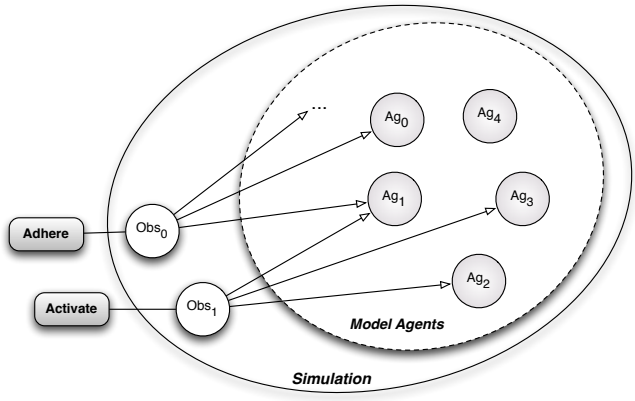

Figure 1: *Observers* inside the simulation space monitor a subset of agents and log necessary information based on their configuration.

An *observer* then extracts group behaviours from the logged data by applying a pattern recognition algorithm. In this paper, we have only relied on clustering, which will be explained in the next subsection.

3.2 Learning and Abstraction

In our prototype, an *observer* logs interaction partners along with the time of the interaction. Once the interaction history $\mathrm{IH}_{\mathrm{Exec}}$ is grown beyond a certain threshold, the *observer* applies a k-means clustering algorithm [12] to find a large cluster C of overlapping interaction partners. When the *observer* finds such a cluster, it infers a generalized group behaviour from the clustered individual interactions by combining their features. The first feature is the set of overlapping interaction partners that are constant for the learned action. Secondly, the *observer* needs to know when and at which rate it should execute the learned action.

The *observer* first finds the time range $[t_{min}, t_{max}]$ of the executed action from all the individual interactions in C. Two cases might happen here: (1) an interaction only occurs within a bound time range, (2) an interaction continuously occurs over time or the *observer* is uncertain whether it has had enough time to determine an upper bound t_{max} of the time range. In order to address the latter case, the *observer* compares the two most recent time stamps an interaction occurs in $\mathrm{IH}_{\mathrm{Exec}}$. If the difference exceeds the observation time, the *observer* sets t_{max} to ∞.

Next, the *observer* extracts the rate of execution defined as the number of interactions in C divided by the number of total computations of the interaction:

$$p_{exec} = \frac{|C|}{|\mathrm{IH}_{\mathrm{Exec}}| + |\mathrm{ihn}|}, \ \mathrm{ihn} \in \mathrm{IH}_{\mathrm{NExec}} \ \& \ \mathrm{ihn}.t \in [t_{min}, t_{max}] \qquad (1)$$

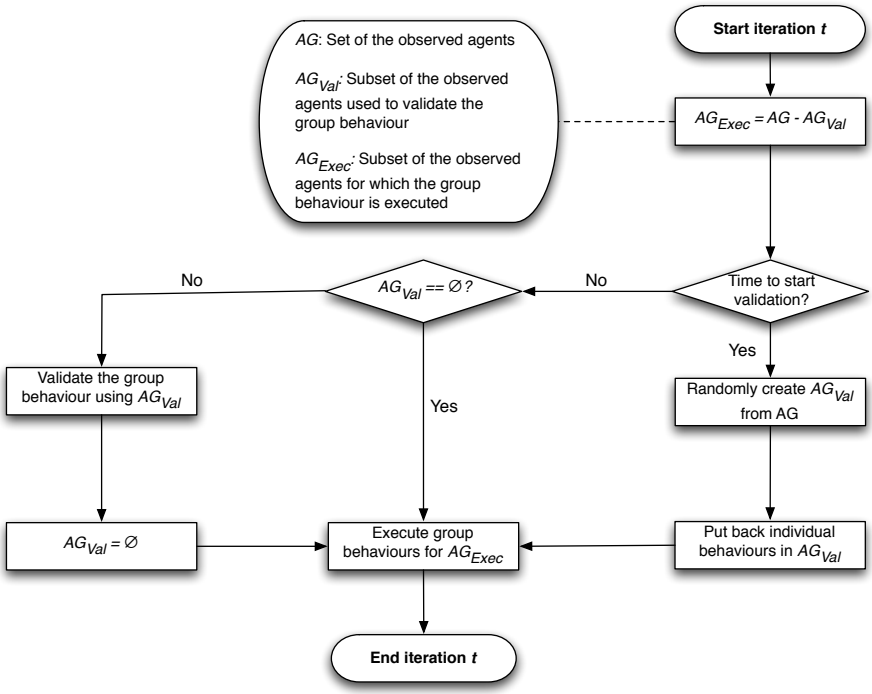

Figure 2: Flow chart of the validation step. At some interval, the *observer* selects a random subset of the observed agents and restores their individual behaviour. The result of their interactions is evaluated in the next iteration to regulate the confidence value. The *observer* continues to execute the group behaviour for all other agents.

For example, all the IH_{Exec} logs in Table 1 except the first log constitute a cluster in which the first column (Ags) is discarded and regarded as wildcard, $[t_{min}, t_{max}]$ is inferred from the last column, and p_{exec} is calculated as described above.

Finally, the *observer* removes *act* from *model agents* and subsumes this action by executing it on behalf of them. If an agent A has the learned action *act* and $A.t \in [t_{min}, t_{max}]$, the *observer* executes *act* for that agent with probability p_{exec}. Since the *observer* has already learned the required interaction partners, it does not need to use the computational resources to find them each time. The *observer* only needs to validate the learned pattern and breaks it down once the learned pattern becomes invalid.

3.3 Validation of the Learned Behaviours

After some time, a learned behaviour might not be valid any more. The *observer* needs to ensure that a learned behaviour is valid to be executed. In order to

monitor the reliability of a learned behaviour, the *observer* assigns an initially unbiased confidence value ($conf_{initial} = 50\%$) to the learned behaviour. At regular time intervals, the *observer* lets some *model agents* execute their original interactions. The confidence value is regulated based on the difference between the actual behaviour of *model agents* compared to the behaviour expected by the *observer* (Fig. 2). In our prototype, we only consider the difference in interaction partners. However, other possibilities like the time at which an individual interaction occurs or the rate at which *model agents* execute their interactions could also be incorporated in the comparison. A confidence measure below a given threshold indicates that a learned group behaviour is not valid any longer and that the *observer* has to restore the *model agents*' original behaviours instead.

4 Experiments

The outlined self-organized optimization method can be employed in arbitrary agent simulations. Biological simulations are particularly suitable applications as biological entities will be directly modelled as agents. When simulating biological systems at the level of inter-cellular and inter-molecular interactions, actions are mostly triggered by collisions or internal agent states. We applied our proposed method to an agent-based simulation of blood coagulation described in the next subsection.

4.1 Model Setup

Blood coagulation emerges from the interplay of various blood factors, i.e. platelets, fibrinogens, serotonins etc. When a collagen protein collides with a platelet, the platelet becomes activated, activated platelets collide with the wound site and secrete several chemicals which in turn activate more platelets in the blood vessel. Gradually, a network of fibrils together with the platelet plug form a clot around the wound site (Fig. 3). An according agent-based model comprises of twelve blood factors modelled as agents. The agents' behaviours are phrased by means of situation-action pairs. There are ten different interactions which fall into two categories: (1) state-dependent interactions, and (2) collision-dependent interactions. The actions themselves introduce local state changes of the agents (represented as internal variables), or they produce or remove agents in the simulation.

4.2 Observer Setup

In our target simulation, there are ten key interactions. Each interaction is monitored by an *observer* that records only the interaction partners. Table 2 lists all the important parameters in our system. Once an *observer* monitors an interaction long enough (t_{wait}), it applies a k-means clustering algorithm to create k clusters. The centroid of the largest cluster is considered to be

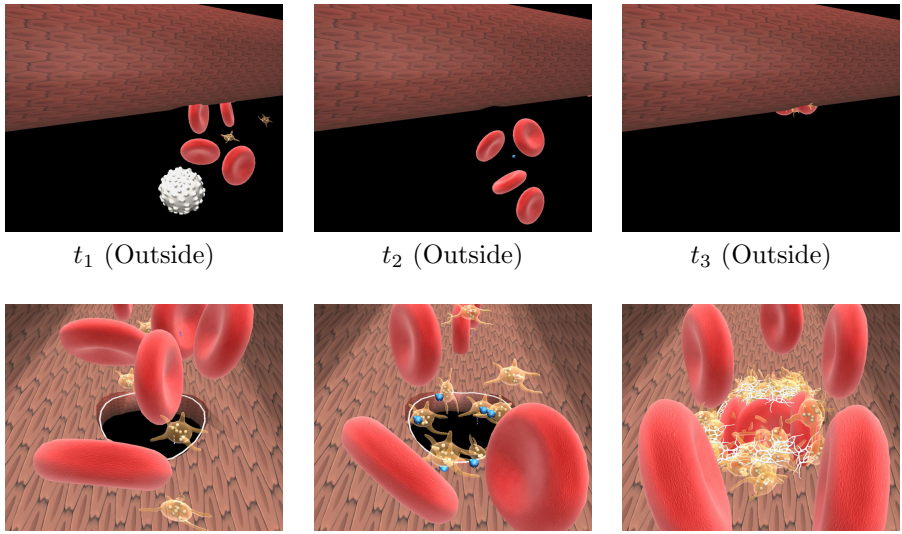

Figure 3: The blood coagulation simulation at different time steps ($t_1 < t_2 < t_3$). The process is observed from two different perspectives: inside and outside of the vessel.

Table 2: System Parameters

Parameter Name	Symbol	Value
Delay before learning	t_{wait}	100
Monitoring interval	$\Delta_{monitor}$	10
Confidence threshold	τ_{conf}	0.3
Number of clusters in k-means	k	30

the learned group behaviour for which $[t_{min}, t_{max}]$ and p_{exec} are inferred. The *observer* subsumes the learned interaction by executing it on behalf of the *model agents*. The *observer* allows some randomly chosen *model agents* to execute their original interaction in predefined intervals ($\Delta_{monitor}$) and validates the result of that interaction with the expected result. The confidence of the learned pattern is regulated accordingly. If the confidence of a pattern is less than some threshold (τ_{conf}), the learned pattern will be removed from the simulation.

4.3 Results

Our proposed approach successfully identified all the group behaviours within the simulation. For example, **Random Walk** is a self-triggering action found to be executed with probability $p_{exec} = 100\%$ and $t \in [0, \infty]$. **Produce Platelet** is an interaction executed in $t \in [3, \infty]$ with probability $p_{exec} = 100\%$. **Activate**

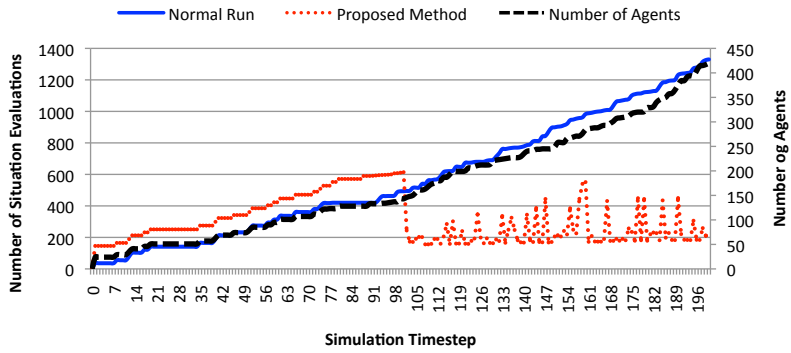

Figure 4: Number of function evaluations and agents over simulation timestep.

is another collision based example with $p_{exec} = 77\%$ and $t \in [90, 95]$.

Figure 4 shows the number of situations evaluated over the simulation time. Due to the model's simplicity, the number of situations to be calculated increases linearly as more agents enter the simulation space. When the proposed abstraction method is utilized, the number of situations to be calculated is kept steadily constant. The difference between the two methods before the abstraction starts ($t < 100$) is due to the overhead of having *observers* which is constant and ignorable. The peaks in our proposed method indicate the validation time in which some *model agents* are allowed to execute their action.

Figure 5 shows the change of confidence value for one of the learned patterns. Since there is no learned pattern before $t = 100$, the confidence value is also 0. However, after the *observer* abstracts an individual behaviour, the confidence value is initially set to 50%. It should be noted that since in this simulation all the actions have been learned and abstracted perfectly correct, the confidence values always increase over time.

5 Conclusion and Future Work

We proposed an approach to reduce the computational complexity of agent-based simulations while maintaining the capacity for arbitrary interactions among agents. In a self-organizing manner, *observer agents* monitor interactions during the simulation to abstract group behaviours once they find an interaction pattern. By replacing individual behaviours with the learned group behaviour, computational costs are reduced. We showed that our approach successfully abstracted all the behaviours in an agent-based simulation of blood coagulation.

We will further extend this work by introducing more levels of abstraction hierarchies that are created automatically upon group behaviours detected by *observer agents*. We are also working to address automatic proliferation of *ob-*

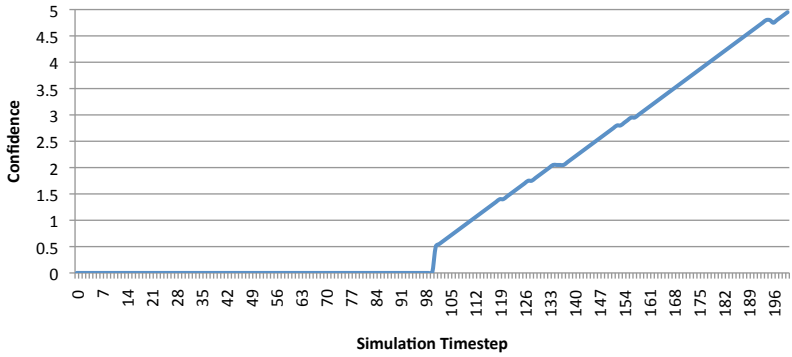

Figure 5: The confidence value over time shown for an exemplarily learned pattern.

server agents, each configured differently to explore a subspace of possible group behaviours. The relation between group behaviours and emergent phenomena is another promising area to be investigated. Although no *a priori* knowledge of high level patterns was incorporated in our design, the possibility to incorporate predefined high-level patterns should be considered. If patterns are described at different scales, multiscale modeling can be restated as finding transitions from low-level to higher-level patterns.

Bibliography

[1] Banzhaf, W., "Artificial chemistries - towards constructive dynamical systems", *Solid State Phenomena* (2004), 43 – 50.

[2] Chen, Chih-Chun, Christopher D. Clack, and Sylvia B. Nagl, "Multi-level behaviours in agent-based simulation: colonic crypt cell populations.", *Proceedings of the Seventh International Conference on Complex Systems*, (2008).

[3] Chen, Chih-Chun, Christopher D. Clack, and Sylvia B. Nagl, "Identifying multi-level emergent behaviors in agent-directed simulations using complex event type specifications", *Simulation* **86** (January 2010), 41–51.

[4] Corning, Peter A., "The re-emergence of "emergence": A venerable concept in search of a theory", *Complexity* **7**, 6 (2002), 18–30.

[5] Denzinger, Jorg, and Michael Kordt, "Evolutionary on-line learning of cooperative behavior with situation-action-pairs", *Proceedings of the Fourth International Conference on MultiAgent Systems (ICMAS-2000)* (Washington, DC, USA,), IEEE Computer Society (2000), 103–110.

[6] Dessalles, J.L., and D. Phan, "Emergence in multi-agent systems: cognitive hierarchy, detection, and complexity reduction part i: methodological issues", *Proceedings of the Symposium in Agent-based Computational Methods in Finance, Game Theory and their applications*, vol. 564 of *Lecture Notes in Economics and Mathematical Systems*, Springer (September 2005).

[7] Dorin, Alan, and Jon McCormack, "Self-assembling dynamical hierarchies", *Artificial life eight* (2003), 423.

[8] Jacob, Christian, and Ian Burleigh, "Biomolecular swarms: An agent-based model of the lactose operon", *Natural Computing* **3**, 4 (December 2004), 361–376.

[9] Kauffman, Stuart, *At Home in the Universe: The Search for the Laws of Self-Organization and Complexity*, Oxford University Press (1995).

[10] Kniemeyer, O., G. Barczik, R. Hemmerling, and W. Kurth, "Relational Growth Grammars—A Parallel Graph Transformation Approach with Applications in Biology and Architecture", *Applications of Graph Transformations with Industrial Relevance: Third International Symposium, AGTIVE 2007, Kassel, Germany, October 10-12, 2007, Revised Selected and Invited Papers* (Berlin, Heidelberg,), Springer-Verlag (2008), 152–167.

[11] Lavelle, Christophe, Hugues Berry, Guillaume Beslon, Francesco Ginelli, Jean-Louis Giavitto, Zoi Kapoula, André Le Bivic, Nadine Peyrieras, Ovidiu Radulescu, Adrien Six, Véronique Thomas-Vaslin, and Paul Bourgine, "From molecules to organisms: towards multiscale integrated models of biological systems", *Theoretical Biology Insights* **1** (2008), 13–22.

[12] MacQueen, J. B., "Some methods for classification and analysis of multivariate observations", *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (L. M. L. Cam and J. Neyman eds.), vol. 1, University of California Press (1967), 281–297.

[13] Prusinkiewicz, Przemyslaw, and Aristid Lindenmayer, *The Algorithmic Beauty of Plants*, Springer-Verlag (1996).

[14] Rasmussen, Steen, Nils A. Baas, Bernd Mayer, Martin Nilsson, and Michael W. Olesen, "Ansatz for dynamical hierarchies", *Artificial Life* **7**, 4 (2001), 329–353.

[15] Salazar-Ciudad, I., "Tooth Morphogenesis in vivo, in vitro, and in silico", *Current Topics in Developmental Biology* **81** (2008), 342.

[16] Sarraf Shirazi, Abbas, Sebastian von Mammen, and Christian Jacob, "Adaptive modularization of the mapk signaling pathway using the multi-agent paradigm", *Proceedings of the 11th international conference on Parallel problem solving from nature: Part II* (Berlin, Heidelberg,), PPSN'10, Springer-Verlag (2010), 401–410.

[17] Schuster, Peter, "How does complexity arise in evolution", *Complex.* **2**, 1 (1996), 22–30.

[18] Servat, David, Edith Perrier, Jean-Pierre Treuil, and Alexis Drogoul, "When agents emerge from agents: Introducing multi-scale viewpoints in multi-agent simulations", *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, Springer-Verlag (1998), 183–198.

[19] Vicsek, Tamás, and Anna Zafiris, "Collective motion", *Reviews of Modern Physics* (Submitted: 2010).

[20] von Mammen, Sebastian, and Christian Jacob, "The evolution of swarm grammars: Growing trees, crafting art and bottom-up design", *IEEE Computational Intelligence Magazine* (August 2009).

[21] von Mammen, Sebastian, Jan-Philipp Stehöfer, Jörg Denzinger, and Christian Jacob, "Self-organized middle-out abstraction", *IWSOS 2011: 5th International Workshop on Self-Organizing Systems*, (2011), 26–31.