Author version of Chapter 6 of "The Computer After Me: Awareness and Self-Awareness in Autonomic Systems".

Bring it on, Complexity!
Present and future of self-organising middle-out abstraction

Sebastian von Mammen[1] & Jan-Philipp Steghöfer[2]
[1]*Institute of Computer Science*
[2]*Institute for Software and Systems Engineering*
Augsburg University

1.1 The Great Complexity Challenge

The inherent complexity of many man-made or naturally occurring challenges—such as understanding the influence of human interference in ecosystems or interacting biological processes—is widely acknowledged. The ubiquitous networking paradigm has highlighted the elaborate webs of interactions and interdependencies between living beings, objects and processes. Yet we still lack an algorithmic framework capable of tackling the complexity of the world in terms of representation and computation. Thus, any step toward understanding—and predicting—the dynamics and emergent phase transitions of complex systems would greatly contribute to the advancement of science. Present-day societal challenges that could benefit from this kind of knowledge are plentiful, and can be found in fields ranging from the life sciences to economics and engineering. To some extent, the mathematical analysis of complex systems can provide some insights about the phase transitions that may occur over time [Haken (1980); Fuchs (2013)]. However, this approach requires a great deal of effort and does not scale well, becoming intractable as the number of factors involved in a system increases.

What is more, the interactions that drive system transitions have to be identified and formalised a priori by the modeller. In contrast, an ideal

model building process should require as little information as possible about a system's actual behaviour. It should be enough to only describe how the parts of a system interact, without building in any assumptions about when feedback cycles might be triggered to snowball into fundamental global system changes. In a model of this kind, the parts of the system that interact according to sets of internal rules (and so without any external, higher-level drivers of their collective behaviour) are known as 'agents'. Each agent in such a model is a self-contained entity with its own individually accessible data, states and behaviours. The sequences of interactions among agents and the traversal of their states in a computational simulation correspond to the emergent feedback cycles and phase transitions of complex systems. If we were able to detect patterns that are precursors to phase transitions and patterns that correspond to the system's global dynamics, we would automatically become aware of emergent phenomena.

Inspired by some of the grand ideas in artificial intelligence, machine learning, and artificial life, we present the SOMO (self-organised middle-out) algorithm, a concept that might contribute to the outlined quest. Its goal is dynamic abstraction, i.e. bottom-up learning given enough training examples and top-down validation to reaffirm or revoke the previously learned concepts. We take this opportunity to present the SOMO concept with an emphasis on its visionary aspects—how the idea could evolve from its most recent conception, its current implementation, towards that desirable, dreamed-about computer after me.

1.2 Self-organising middle-out abstraction

Early 2011 we presented the self-organised middle-out (SOMO) concept [von Mammen *et al.* (2011)], an approach that automatically builds abstractions bottom-up and validates and revokes them top-down—possibly both at the same time but in respect to different model aspects. As it works in both directions and as it bridges the gap between the orders of the model, it can be considered to operate at the 'meso' level of analysis.

Its foundation is an unsupervised learning method that observes and learns processes which occur—that is to say, emerge—during a computational simulation. A learned process pattern provides a shortcut to driving the evolution of the simulation. Instead of considering the series of all conditions that lead to the process' changes one step at a time, it suffices to recognise the emergence of the process. As a consequence, the detailed

interactions are no longer executed but, whenever the according preconditions hold, the observed side effects are enacted in the system. Such automatically learned patterns may also be understood as abstracted process descriptions and they hold the promise of helping us to understand, explain, and compute complex phenomena in simple terms.

SOMO observes the simulation data and identifies process patterns, 'biased' only in terms of its representations (meaning that the way interaction patterns are represented by SOMO can influence the kinds of patterns that can be detected and so bias the result). The identified patterns are used to refine the computational model that drives the simulation process being observed. As the SOMO algorithm continues to observe and learn the patterns that emerge from the simulation, it continually increases the model's level of abstraction by introducing hierarchies of abstracted patterns. It is hoped that such hierarchies will to some extent coincide with the real-world conceptual boundaries that we identify in natural systems, such as the subdivision of the organisational complexity of animal anatomy into cells, tissues and organs. Since such abstractions are inevitably subject to noise and unknown conditions, we also introduce a confidence measure that is associated with each abstraction.

In the next section (Sec. 1.3), we present a variety of concepts that are both inspiring the SOMO algorithm and closely related to it. Section 1.4 introduces a (borrowed) example that nicely illustrates the emergence of high-order physiochemical compounds. Based on this example, we outline the SOMO concept in Sec. 1.5. Current SOMO implementations are explained in Sec. 1.6 and futuristic implementations around it are presented in Sec. 1.7. In Sec. 1.8, we conclude with a short vision about SOMO's potentials.

1.3 Optimising Graphics, Physics & AI

Various research interests and complementary research trends have been driving the design of the SOMO concept:

- There is the concept of *emergence* that tries to capture novel properties and descriptions of (sub-)systems of higher orders [Baas and Emmeche (1997)].
- There is the need for integrative approaches to *representing, modelling and simulating multi-scale systems*—this challenge is currently addressed by passing up and down value sets from sepa-

rate, sometimes fundamentally disparate, model components [Eissing *et al.* (2011); Horstemeyer (2010)].

- And, there is the need for *abstraction*: a model so comprehensive as to span several degrees of scale, to host a large body of systems and subsystems, and to independently consider their intricate behaviours quickly outmatches the computing capacities of even the greatest of supercomputers.

Abstraction is not only the essence of model building in the first place but it is also the key to expressive and efficiently solvable models. We postulate that a model should be as detailed and as comprehensive as possible, while its (numeric) utilisation for the purpose of rather specific predictions or simulations should automatically lead to model simplifications and abstractions. Whenever possible, this should happen without jeopardising the model validity; whenever necessary, the loss of accuracy the abstractions cause should be made transparent. SOMO pursues this endeavour by building and maintaining hierarchies of abstractions learned from observation. The higher the level of hierarchy, the fewer interactions have to be tested. Such tests are typically intertwined with expensive condition queries—only the state changes of the simulation will be performed to drive its evolution.

Similar shortcuts by means of hierarchical organisation have been conceptualised and implemented in numerous other contexts. For instance, different levels of detail (LOD) of computer graphics resources such as meshes (differing in the numbers of vertices) and textures (differing in the numbers of pixels) are typically organised in hierarchies to allow for fast access to the most commonly used assets, whereas the graphics scenes themselves are often subjected to spatial partitioning hierarchies that allow algorithms to quickly determine which graphics objects need to be rendered in a given view port [Möller *et al.* (2008)].

There is a significant overlap between these *culling* techniques and mechanisms to speed-up the detection of collisions between geometric objects, one of the foundational functionalities of physics engines—both rely on the quick discovery of objects at specific locations. In general, the locations of the geometries may change, which is why the spatial partitioning hierarchies are dynamically created and adjusted. Dynamic adjustments of the bounding volume hierarchies are also required if the geometries themselves are dynamic, for instance if they change their scale. In this case, a method has been shown to yield rather good results that updates the upper half of the hierarchy bottom-up if one of the geometries changes. The lower half

is only updated selectively in a top-down fashion, as soon as the changed geometry is accessed [Larsson and Akenine-Möller (2001)].

Hierarchical optimisations have also been deployed in the field of artificial intelligence. For example, costly automated planning routines can be pruned early, if high levels of a hierarchy reflect the adherence of a plan's most critical variables [Sacerdoti (1974)]. Similarly, reflective agents need to plan their coordination—hierarchical abstractions of their interaction partners may increase their decision performance, too [Durfee (1999)].

1.4 Emergence and Hierarchies in a Natural System

In Rasmussen *et al.* (2001), an approach, or "Ansatz", to capturing the emergence of physicochemical compound objects with according emergent properties is described. We want to use their example to illustrate the mechanics of SOMO. In their experiments, attracting, repelling, and bonding forces among charged monomers and water molecules are shown to result in higher-order polymer and micelle formations—at each level, the resulting compounds obtain novel physical and chemical properties. In the model, hydrophobic monomers bind to hydrophilic monomers as well as to polymerised hydrophobic monomers, which results in 2^{nd}-order amphiphilic polymers which, in turn, aggregate in 3^{rd}-order micelle structures. At each stage, the resultant compounds exhibit properties different from the underlying constituents; The aggregating nature of the process yields compounds of greater size but it also leads to varying qualitative, geometric structures and differentiated physiochemical behaviours. An adapted illustration of the emergent process is shown in Figure 1.1.

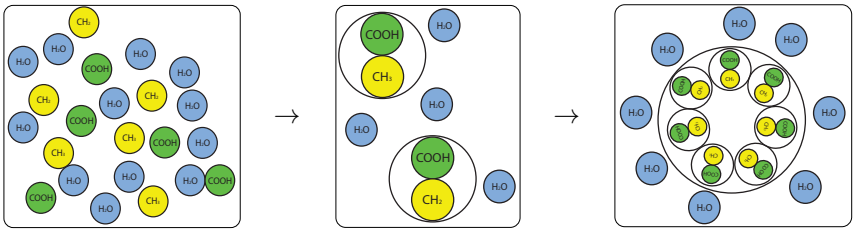

Fig. 1.1 (a) Hydrophobic and hydrophilic monomers immersed in water. (b) Polymers emerge as hydrophobic monomers bind to hydrophilic monomers. (c) A micelle-like structure forms based on aligned polymers with hydrophobic heads and hydrophilic tails. These illustrations are adapted from Rasmussen *et al.* (2001).

The higher-order objects form based on the interactions and (emergent) relationships among the axiomatic objects of the given model. Often, higher-order objects can be captured as spatial aggregations but in general they should be regarded as networks of arbitrarily complex topologies. In accordance with Baas and Emmeche (1997), the authors also stress that emergent characteristics of a (sub-)system are observable in terms of its interactions.

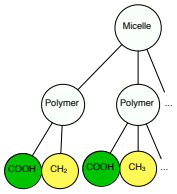

Fig. 1.2: Order hierarchy.

We reflect the subsumption of individual elements by emergent entities of greater order in a hierarchical structure. In the given case, polymers are built from monomers and aggregate to form micelle-like structures (Fig. 1.2). As Rasmussen *et al.* suggest, an observer needs to identify the emerging units and their emergent properties [Rasmussen *et al.* (2001)]; In our approach such observers are immersed in the simulation and observe the state and interaction patterns of the model entities. The observers further simplify the entities' computational representations in accordance with the learned behavioural patterns. Individual entities and their behaviours are subsumed by higher order entities that perform the learned patterns only in order to prune the computational complexity. However, we do not postulate a *necessary* coincidence between the learned high-order entities and emergent entities that we ourselves would identify, as in the micelle-example. Rather, we assume that there is a great chance that the learned patterns and the ones recognised by humans overlap to some extent—it is possible that the human-identified orders represent all but a small fraction of the automatically generated abstractions. In order to clarify this distinction, we step through an exemplary run of the SOMO algorithm in the next section, using the self-assembly of micelles as a running example.

1.5 The Technical Concept of SOMO

For our approach, we consider the elements of a model agents, described by their states and behaviours (for a more in-depth formalisation of the agent concept, consider, for instance, Denzinger's generic agent definition [Denzinger and Winder (2005)]). In our example, we distinguish between freely moving reactive agents that represent molecular compounds (similar to artificial chemistries [Dittrich *et al.* (2001)]) and the environment they are immersed in. In particular, in our running example, a large number

of hydrophobic and hydrophilic monomers is immersed in an aqueous environment. With the beginning of the simulation, the agents start to interact with each other and with the environment based on their behavioural rules. Together with the initial configuration of the system, these rules determine the result of the simulation, and if correctly phrased, they would result in the emergent phenomena described in the previous section.

1.5.1 *Observation of Interactions*

In addition to the model agents comprising molecular compounds and the environment, the SOMO concept introduces *observer* agents that monitor the interactions of the model agents as well as the conditions under which they occur. In the context of the micelle-forming example, observers do not have to make assumptions about the model agents' internal states and behaviours—only their actually triggered, externally observable state boundaries (i.e., the observed boundaries of the domain over which the state variable is defined) and state changes are relevant. However, the potency of the observers can be increased by granting them access to the agents' behavioural rule sets, to their internal states, and, thus, to their activated rules[1]. Following the fundamental concept of cause and effect, the observed interactions are recorded in terms of states and state changes. States that lead to certain state changes are translated into boundary conditions, or *predicates*, whereas state changes simple describe the transition from one state attribute value to another. Boundary conditions of time (i.e., the agents' timing), proximity between agents, or their mere presence or absence come to mind. Conjointly occurring pairs of boundary conditions and state changes are stored in *interaction histories* over a certain period of time. A sliding time window reduces the storage required and lets the observers "forget" rare or singular events.

In the example, a pair of hydrophilic and hydrophobic monomers may attract each other, then stick together. A strong correlation between their locations would emerge, quickly resulting in a static relationship between their position states. The molecules might stick together over a long period of time. An observer would identify this behaviour and infer from the observations that these molecules will, under the given conditions, continue to stick together. Therefore, instead of continuously adjusting their

[1]Focusing on the observation of state changes deems simpler than considering the underlying, responsible behavioural representations, as those would have to be correctly interpreted and related to the simulation context by an external observer.

locations based on their proximities at each time step of the simulation, an abstraction is introduced into the model: For now, the monomers are considered constantly attracted, or *bonded*. These bonded monomers, or *polymers*, are likely to aggregate in a *micelle*-like organisation because of the interplay with the aqueous environment: The polymers' heads align to face the water molecules, whereas their tails avoid them. Again, this formation is recognised and learned by the SOMO observers.

Instead of using specialised observers, the agents that make up the model can themselves observe interactions and the environment. In many cases, however, it is desirable to separate SOMO logic and the simulation model to maintain a clear distinction between the behaviours of the automatically learned abstractions and the original model. Independent of the kind of agent that takes on the task of observation and abstraction, the observers are subjected to certain restrictions. First, they are subject to an "event horizon", i.e., they do not perceive the entire system but only portions of it. This is due to the fact that an omniscient observer would have to deal with a vast amount of data, nullifying the scalability benefits SOMO was designed for and making it necessary to introduce limits of the observations. Second, even though observers make no assumptions about the model of an observed agent, they are limited to knowledge they have been granted access to—they can only perceive states and state changes they were designed to sense. Therefore, if interactions take place hidden from the observers, for instance direct messaging between agents based on hidden internal states, these interactions will not become part of the interaction history.

These restrictions bias the abstraction process. If the scope of the simulation is well-defined, these restrictions can be mitigated rather easily—the SOMO agents can be distributed across the interaction space to cover important "hot spots" and the system designer can ensure the agents' ability to observe all relevant states and state changes. For more ambitious projects, however, it might be necessary to create a wide variety of observers, capable of identifying many different kinds of interactions.

Heterogeneous configurations are also possible. For instance, a subset of agents might be part of the original model and yet observe and abstract others, whereas the remainder of the agent population might be either model agents or observer agents. Naturally, hybrid agents, that play both roles, are useful, if an abstraction hierarchy is part of the model. In the following, in order to avoid additional case distinctions, we will only distinguish between (1) a strict separation between observer and model agents, and (2)

the capacity of all agents to observe and abstract.

1.5.2 *Interaction Pattern Recognition and Behavioural Abstraction*

The entries of the interaction history not only comprise some anonymous information about states and subsequent state changes but they also reference the involved interaction partners. Similar to [Ramchurn *et al.* (2004)], we use the interaction histories as databases for finding patterns in the agents' interaction behaviours. Previously unknown patterns, or *motifs*, can be identified in time series relying on various techniques such as learning partial periodic patterns [Han *et al.* (1999)], applying efficient, heuristic search [Chiu *et al.* (2003)], online motif search [Fuchs *et al.* (2009)], and even the identification of patterns of multiple resolutions [Wang *et al.* (2010)]. Motif detection is adapted to interaction histories by assigning symbols, e.g., A or B, to specific log entries and finding patterns in the resulting strings, e.g., BBABCCBBABDA. In the given example BBAB is a motif candidate.

The recurring sequence of interactions contained in the motif as well as the conditions that are part of it can be the basis for a *behavioural abstraction*. If interactions are recognised repeatedly, they can be abstracted in several ways—most simply, the predicates are not always checked; in full glory, a complex sequence of interactions can be fully abstracted and only the aggregated side effects, i.e. the state changes, can be enacted in the system. Hence, a motif that provides comprehensive information about the interaction partners and the actual interactions, would allow to rewrite the agent rules as efficient sequences of unconditional instructions, with source and target agents readily in place.

A repeatedly occurring motif in the example system is the interaction between hydrophilic head and hydrophobic tail of a polymer. As the effect of this interaction stays the same once the monomers have bonded, it is not necessary to check these conditions and calculate the result of the interaction any longer. An observer that has monitored this interaction can thus suspend the rules that cause the effect but rather enact it directly. Of course, such an intervention requires direct access to the agents' rule bases and might not be possible in some systems (cf. Sec. 1.5.5). Instead of suspending a specific agent's behavioural rules directly, it's possible to subsume the agent as a whole. The next section will shed more light on this approach of hierarchical agent subsumption.

1.5.3 *Creating and Adjusting Hierarchies*

The polymer formation from simpler monomers provides an example for an abstraction even more powerful than simplifying specific interaction rules: If the agents keep interacting in a predictive manner, among each other and with their environment, they can be subsumed by one *meta-agent* that takes their place and that exhibits their external behaviour without continuously (re-)evaluating the interactions of its constituting elements. Recursive subsumption of agents and meta-agents yields a hierarchy of ever more abstract meta-agents.

The formation of hierarchies can be implemented by means of a set of special operators. In order to establish a hierarchical relationship, an agent might *enter* another agent. Alternatively, it might be *adopted* by another agent. Both actions yield corresponding parent-child relationships between the two agents. Such a parent-child relationship is reverted by *raising* a child in the hierarchy.

Depending on whether the agents observe their own interaction histories or specialised observers are used in the system, different kinds of behaviour are possible:

- If an agent observes its own interaction history and detects that it constantly interacts with another agent (or a group of other agents), it can create a new agent, assign its own abstracted behaviour, enter this new agent and deactivate itself. The newly created higher order agent then adopts all other agents that formed the original behaviour, adding their abstracted behaviour to its own, and deactivating them as well.
- If specialised observers are deployed in the system, they create the meta-agents and assign the agents to be subsumed to them. The meta-agent then follows the same steps as above.

The end result in both cases is a meta-agent that behaves just like the group of agents to the outside but does not need to evaluate internal interactions. The polymer as well as the micelle are examples of structures that can be abstracted in this fashion. In fact, the micelle shows how a true hierarchy can form: in the course of the simulation, polymers form first, are detected by the observers, and abstracted. Then, the polymers form a micelle which is internally stable and behaves consistently towards its environment. It can thus again be detected and abstracted so that only interactions between the micelle and the water molecules have to be

evaluated.

Repeated applications of these abstraction rules yield continuously growing hierarchies with increasingly simplified behaviours. At the same time, hierarchies are dissolved when no longer appropriate. For this purpose, meta-agents repeatedly check for validity of the abstraction they represent by checking whether the original predicates still hold or by temporarily disbanding the abstractions, checking for the occurence of the abstracted interactions and either re-abstracting or abandoning the abstraction.

The subsumption of agents and their behaviours closely resembles the concept of modularisation and crafting hierarchical code. Figure 1.3 shows an according visual programming perspective on agents, their behaviours and behavioural interrelations; individual operators (spheres) are recursively nested to allow for the hierarchical design of behavioural modules, whereas the connections between inputs and outputs (cones) determine the flow of information at each hierarchical level [von Mammen and Jacob (2013)]. The realisation of this visual modelling language has, in parts, been motivated by the need of a generic, hierarchical representation of agent behaviours.

1.5.4 *Confidence Measures*

The identification of motifs in the interaction history as well as the decision to resolve a hierarchy are based on *confidence estimation*. There is a large body of work around confidence in statistics [Kiefer (1977)] and its effective standardisation for use in the natural sciences is a vivid research area [Louis and Zeger (2009)]. Confidence measures are also used in computational models of trust [Kiefhaber *et al.* (2012)]. The general idea is to estimate the probability that a pattern occurs based on its preceding frequency over a given period of time.

In SOMO, repeated observation of interaction patterns increases the confidence value. A sufficiently great confidence value leads to abstraction. The confidence value also determines the abstraction's lifespan. Confidence metrics that are too generous, i.e., that cause too long abstraction lifespans, diminish the accuracy of a simulation. Abstracted behaviours are repeatedly checked for validity by either exposing the subsumed agents to the environment and observing their behaviour again or by checking the predicates that have been identified in the abstraction process. This check can occur at fixed time intervals, at the designated end of the meta-agent's lifespan, or based on heuristics such as the degree of activity in its local en-

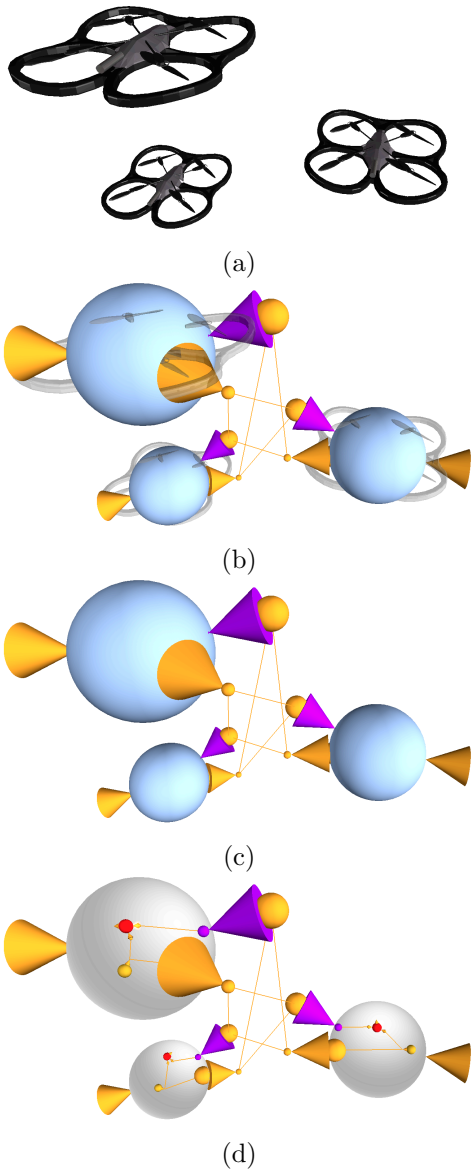

Fig. 1.3 (a) Three quad-copter agents situated closely together. (b) Projection of the agents' behavioural operators and their interrelations into the agent space. (c) Focus on the behavioural network. (d) Introspection of the agents' behavioural modules reveals hierarchically nested, lower-level operators and their connectivity.

vironment. If the abstraction proves valid, confidence rises and the checks become less frequent. However, if the abstraction proves invalid, confidence sinks and the abstraction is either checked more often or abandoned completely.

In case of miscalculations, the simulation could be reset to a previous simulation state, adjusted and partially recomputed. This additional overhead might make it hard to reach a gain in efficiency. On the other hand, if confidence is assigned too cautiously to motifs, abstraction hierarchies do not get a chance to form in the first place. Thus, a careful balance has to be found. Learning methods as introduced in Sec. 1.5.6 can help find suitable parameters for concrete scenarios.

1.5.5 *Execution Model*

Our stated goal is to create a learning abstraction mechanism that makes as few assumptions as possible about the agents it is working with. However, in order for behavioural abstraction and hierarchical abstraction to work, the underlying execution model has to fulfil some requirements.

As mentioned before, *behavioural abstraction* requires that some of the internal rules according to which an agent operates can be suspended by an external entity. This is a natural assumption if agents observe themselves or if they can issue the rule's temporary removal (e.g., to a global simulation engine). However, if the agents are fully opaque and abstraction is performed by specialised observers, they need to be able to influence them directly. As the system designer usually has complete control over the simulation environment, it should be possible to implement such a feature within the environment directly.

For *hierarchical abstraction*, we assume that execution of the agents follows the hierarchy as well. First, root nodes are considered for execution. Their children are considered recursively, only if they are active, i.e. if they are not suspended. Deactivating child nodes instead of removing them from the simulation entirely is necessary in order to check the abstractions' validity. Their (inactive) maintenance as part of the simulation hierarchy also serves to update their states as part of abstracted high-level behaviours.

Since simulations are usually closed systems, it is safe to assume that a benevolence assumption holds. This means that no agent in the system has an incentive to deceive the observers and information about states and state changes is provided freely and without inhibition.

1.5.6 *Learning SOMO: Parameters, Knowledge Propagation, and Procreation*

As an unsupervised learning approach, the self-organised middle-out learner will have to be able to learn about itself and thus become self-aware in a sense. A simple example is the requirement to learn which abstractions worked in the past and which failed to show the desired benefits. If abstractions had to be quickly dissolved, the SOMO observer that created them obviously did something wrong. Either its observations were faulty or the parameters were sub-optimal, e.g., the confidence value that is used to estimate when it is safe to assume that an interaction actually occurs repeatedly.

On the other hand, multiple SOMO observers deployed in the system should be able to learn from each other. An abstraction that has proven valid for one observer should not have to be learned by other observers in the system. Instead, patterns should be propagated and the knowledge acquired should be spread throughout the system. This way, the SOMO learner becomes an organic, learning, improving system within the system that constantly revises and improves its knowledge about the environment and itself by the meta-interaction of the individual observers.

Thus, SOMO agents learn on two levels: they adapt and improve their individual learning and abstraction parameters to become well suited for the niche they occupy in the simulation; and they exchange knowledge with each other and incorporate this knowledge in their decision making process.

The former kind of learning can be performed based on the data the agents collect and based on the perceived results of the actions performed by the agents. If a behavioural abstraction has proven unstable, the agent can, e.g., increase the confidence value at which it abstracts behaviour. It would thus have to be more certain that a behaviour occurs repeatedly in the same fashion before abstracting it. More excitingly, however, an additional feedback loop can be added to a SOMO learner that uses the data collected by the agent to simulate different sets of parameters and the results they would have yielded. Such a simulation-within-the-simulation can use an evolutionary algorithm (EA) to evolve and test a population of parameter sets, simulate the learner's behaviour and use a fitness function that checks whether the parameters would have found abstractions that have actually proven valid. A parameter set with a high number of valid abstractions gets a higher fitness value and may be adopted. The EA can run concurrently and change the parameter settings whenever better results

are obtained than possible with the current parameters. A similar approach has, e.g., been used to create and simulate new traffic light switching rules in a traffic-control scenario [Prothmann *et al.* (2011)].

The latter kind of learning, in which patterns, motifs, parameter sets, etc. are propagated in the system can be implemented using gossiping algorithms [Eugster *et al.* (2007)]. These consensus approaches are built around local communication in which information is primarily exchanged with neighbours, aggregated, and spread through the system. As the communication is limited to a small number of agents, the system is scalable and since information is always disseminated along several trajectories, the system is robust. A major concern in the design of such algorithms is "eventual consensus", i.e., ensuring that at one point, all agents have access to the information. Fortunately, the SOMO learning approach does not have this requirement as even local knowledge exchange can improve its efficiency and thus, relatively simple gossiping protocols can be used.

Whenever a SOMO agents learns a new set of parameters, a new motif, or that a certain abstraction has proven valid, it can provide this information to other agents in its neighbourhood. These recipients can elect to use this information, e.g., because they are situated in a similar environment, or discard them. They can also elect to augment or redact the information and send them on to their own neighbours. This way, knowledge spreads through the system and allows the learning agents to profit from the experiences of others. Similar techniques have, e.g., been used to spread reputation information in multi-agent systems [Bachrach *et al.* (2009)].

For the transmittal of information between SOMO learners, a language for the knowledge of the agents has to be defined. Apart from using it in the exchange of information, it can also be used to store the knowledge between simulation runs. This way, different runs of the same simulation can profit from knowledge learned previously and—if the simulations are similar enough—different simulations can re-use knowledge learned previously. A SOMO learner that is repeatedly used in the same setting can thus evolve along with the simulation and improve over time.

In settings in which the simulation is highly dynamic, an additional meta-learning approach can be used. At the start of the simulation, SOMO learners are spread evenly within the simulation space. If a SOMO agent finds itself in a highly dynamic environment, with many entities to observe and many interactions, it can procreate by spawning a duplicate of itself. This new agent carries the same knowledge as its father and can become

active in the same area. Thus, the SOMO system self-organises towards a structure in which learning takes place in those locations where it is most beneficial and where most interactions occur.

While the outlined meta-learning approaches should improve SOMO's ability to find valid abstractions and simplify the simulations, they incur additional computational cost as well as increase the memory requirements. Therefore, the use of these faculties has to be evaluated carefully for each new simulation setting and the trade-off between the resources required for meta-learning and the benefit has to be analysed.

1.6 Current implementations

In several publications, Sarraf Shirazi et. al present the exploration and extension of SOMO implementations in the context of biological simulations [Sarraf Shirazi *et al.* (2010, 2011a,b, 2012 (in press); Jacob *et al.* (2012); von Mammen *et al.* (2012); Sarraf Shirazi *et al.* (2013 (submitted)]. Therein, the application domain slightly shifted from protein-interaction networks (in context of the MAPK signalling pathway) towards cell-cell/cell-membrane interaction systems (in context of blood coagulation processes). More importantly, the model representations underwent an evolution as well: Sarraf Shirazi and his colleagues (one of them is an author of this chapter, S. von Mammen) first learned clusters of intertwined functions of gene expression rates by correlating their results—initially by means of artificial neural networks, then by means of genetic algorithms. The second iteration of implementations featured rule-based multi-agent representations and sets of learning observer agents that logged and subsumed the activities of the agents in the simulation. For instance, blood platelets and fibrinogens that are stuck together are subsumed by meta-agents with reduced rule sets and which represent the blood clot.

Current SOMO implementations have shown the effectiveness of the concept. In early experiments the number of tests performed as part of the simulation was successfully reduced. In later experiments, Sarraf Shirazi et al. were able to show that the overall performance, also considering the computational overhead needed for observing and dynamic learning, can be improved.

The original SOMO concept foresees the possibility to expose SOMO agents without prior knowledge to an arbitrary multi-agent simulation to automatically infer hierarchies of patterns from the observed processes. In

order to reach this desirable goal, numerous challenges still have to be addressed. The universal deployment of the SOMO concept requires, for instance, a generic learning mechanism for identifying arbitrary patterns (e.g., learning classifier systems [Wilson (1995)]), a universal approach to measuring and comparing confidence values and an accordingly tuned reinforcement learning mechanism, as well as a comprehensive formalisation of representation and algorithms.

An example of a meso-level abstraction algorithm with a more technical focus has been presented by Steghöfer *et al.* (2013) with the HiSPADA algorithm. The Hierarchical Set Partitioning Algorithm for Distributed Agents forms abstraction hierarchies within an agent society based on scalability metrics. If an agent system solves a computationally intensive problem that is defined by the individual agents (such as scheduling in power management scenarios) and that can be hierarchically decomposed, intermediaries can be introduced to solve parts of the original problem. Each intermediary solves a sub-problem that is defined by the agent it directly controls. The runtime of the problem solver depends on the number of agents controlled by an intermediary. If it exceeds a certain threshold, an additional layer of intermediaries can be introduced to divide the controlled agents. An intermediary acts as a black box to the outside, much like the meta-agents in the hierarchical abstraction. However, the intermediary is not the result of a learning process based on the interaction patterns of the agents but merely a result of an internal constraint violation. Nevertheless, the concept has proven to improve scalability in large systems and provides a starting point for future research.

1.7 Awareness beyond virtuality

It has already been shown that current implementations of SOMO are capable of pruning computational complexity in multi-agent based simulations and identifying emergent processes. A broadly deployable, unbiased SOMO implementation would make it possible to compute models with large numbers of approximate constants, such as in our perceived reality. This would make it possible to integrate vast quantities of scientific facts, across all levels of scale and scientific disciplines, for consideration in simulations.

1.7.1 *Integration & emergence*

The result would be *virtually* unlimited computing power for models with large numbers of approximate constants—as in our perceived reality. Vast amounts of scientific facts, across all levels of scale and scientific disciplines, could be integrated for consideration in simulations. The development of an organism could be computed bottom-up from a single fertilised cell. As we believe SOMO to be principally capable of developing awareness for previously unknown emergent phenomena—both *in-silico* and *in-vivo*—the organism's systems would be identified automatically. The recognised patterns are expressed in algorithmic rather than traditional mathematic representations, and therefore human-readable and comparable to human reasoning.

1.7.2 *Model inference*

What is more, the SOMO concept need not be limited to virtual simulations. Heuristic learning methods could supply feasible solutions for gaps in theories, for which empiric researchers haven't provided answers yet. However, instead of limiting SOMO to virtual simulations, it could operate on top of a smart sensory network (SOMO net), an advanced wireless sensor networks (cf., e.g., [Akyildiz *et al.* (2002)]). Enhancing SOMO sensory nodes with effectors would further introduce the capability of self-directed inquiry. At this point, the SOMO net could turn into a self-reflective machinery similar to the one developed by Lipson and Pollack (2000) that also grew, the other way round, into a system to automatically infer complex, non-linear mathematical laws from data sets by avoiding trivial invariants [Schmidt and Lipson (2006)]. SOMO net enhanced in this way would be able to autonomously perform observational analysis and pro-active investigations to further accelerate the generation of comprehensive and accurate scientific models.

1.7.3 *SOMO net*

In addition to the sensory functionalities present in a subset of nodes of the envisioned SOMO net, all the nodes would have to provide a runtime environment for a SOMO agent. To begin with, the initialised, *networked* SOMO agent—a conceptual descendant of the SOMO observer as deployed in virtual simulation environments—would sense and transmit data to its neighbours and, in turn, aggregate any received information. The analogies

to distributed learning approaches are obvious, especially in the context of wireless sensor networks [Predd *et al.* (2006)]. However despite the common notion of a global learning task, distributed data sources, and efforts to fuse the aggregated data, the SOMO reaches further.

Quickly, a SOMO agent would learn patterns in the sensed and received, transmitted data and refine its sensing configuration and communication connectivity based on the greatest information gain: it would direct its inquiries to areas of interest, i.e., sensor ranges or nodes that provide (from its perspective) unpredictable information. Depending on the confidence values associated with the learned patterns, the original data sources would be queried once in a while in order to test the abstractions' validity.

As the learned patterns would reference the learning context, i.e., the network location and connectivity of the learning agent, the abstracted information can be passed down the network, enriching the other agents' data bases, without causing confusion. Whenever possible, patterns could be subsumed in higher level abstractions, the validation process stretching across the network.

The self-organised, decentralised learning and validation algorithm would ensure that the system under observation is described at several levels of abstraction, based on the input on numerous nodes with their individual perspectives. At the same time, it would ensure that the processing and communication costs of the networked nodes is minimised—which is of crucial importance for the efficacy and longevity of a wireless sensor network.

1.7.4 *SOMO after me*

SOMO and SOMO nets would make correlations between processes apparent that have never been thought of before. These new insights, could, due to the immense complexity that SOMO promises to handle, help to build sustainable, progressive, evolving economic and ecological infrastructures for the great challenges of human kind.

At the same time, accessible methodologies for large-scale data modelling and exploration would become an (even more) important limiting factor. In order to counter this arising challenge, we have been developing INTO3D, an integrated visual programming and simulation environment [von Mammen and Jacob (2013)]. Combined with SOMO's computing abilities such environments could make model-building and simulation feasible and attractive to non-scientists, or rather, they could turn anyone into a

scientist and revolutionise everyday life.

1.8 The future of SOMO

In summary, the SOMO algorithm and SOMO nets hold the promise of revealing hitherto unsuspected correlations between processes. Such new insights, and the immense complexity that SOMO can handle, would help to build the sustainable, progressive and evolving economic and ecological infrastructures for tackling the major challenges humankind faces today. Our current work on SOMO is focused on pattern detection in observed interactions and the possibilities for propagating knowledge about abstractions through the system. Once the implementations of SOMO have reached maturity, we envisage that research can shift to analysing how the learned abstractions and features correlate with the behaviours we find in higher order emergent phenomena. Whether we will find striking similarities, or instead discover these to be two completely different forms of complex systems, remains an exciting open question at this time.

Bibliography

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y. and Cayirci, E. (2002). Wireless sensor networks: a survey, *Computer networks* **38**, 4, pp. 393–422.

Baas, N. and Emmeche, C. (1997). On emergence and explanation, *Intellectica* **25**, 2, pp. 67–83.

Bachrach, Y., Parnes, A., Procaccia, A. D. and Rosenschein, J. S. (2009). Gossip-based aggregation of trust in decentralized reputation systems, *Autonomous Agents and Multi-Agent Systems* **19**, 2, pp. 153–172.

Chiu, B., Keogh, E. and Lonardi, S. (2003). Probabilistic discovery of time series motifs, in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (ACM), pp. 493–498.

Denzinger, J. and Winder, C. (2005). Combining coaching and learning to create cooperative character behavior, in *Proc. IEEE Symposium on Computational Intelligence and Games* (IEEE), pp. 78–85.

Dittrich, P., Ziegler, J. and Banzhaf, W. (2001). Artificial chemistries - a review, *Artificial Life* **7**, pp. 225–275.

Durfee, E. H. (1999). Practically coordinating, *AI Magazine* **20**, 1, pp. 99–116.

Eissing, T., Kuepfer, L., Becker, C., Block, M., Coboeken, K., Gaub, T., Goerlitz, L., Jaeger, J., Loosen, R., Ludewig, B., Meyer, M., Niederalt, C., Sevestre, M., Siegmund, H.-U., Solodenko, J., Thelen, K., Telle, U., Weiss, W., Wendl, T., Willmann, S. and Lippert, J. (2011). A computational systems biology software platform for multiscale modeling and simulation: integrating whole-body physiology, disease biology, and molecular reaction networks, *Frontiers in Physiology* **2**, 1–10.

Eugster, P., Felber, P. and Le Fessant, F. (2007). The "art" of programming gossip-based systems, *SIGOPS Oper. Syst. Rev.* **41**, 5, pp. 37–42, doi: 10.1145/1317379.1317386, URL `http://doi.acm.org/10.1145/1317379.1317386`.

Fuchs, A. (2013). *Nonlinear Dynamics in Complex Systems*, chap. Self-organization and Synergetics (Springer Berlin Heidelberg), ISBN 978-3-642-33551-8, pp. 147–157, doi:10.1007/978-3-642-33552-5{_}8, URL `http://dx.doi.org/10.1007/978-3-642-33552-5_8`.

Fuchs, E., Gruber, T., Nitschke, J. and Sick, B. (2009). On-line motif detection

in time series with swiftmotif, *Pattern Recognition* **42**, 11, pp. 3015 – 3031, doi:DOI:10.1016/j.patcog.2009.05.004, URL `http://www.sciencedirect.com/science/article/B6V14-4W99VVS-1/2/422d19cac31c5f2f5d47a1567af4fb60`.

Haken, H. (1980). Synergetics, *Naturwissenschaften* **67**, 3, pp. 121–128, doi:10.1007/BF01073611, URL `http://dx.doi.org/10.1007/BF01073611`.

Han, J., Dong, G. and Yin, Y. (1999). Efficient mining of partial periodic patterns in time series database, in *Proceedings of the International Conference on Data Engineering* (Citeseer), pp. 106–115.

Horstemeyer, M. (2010). Multiscale modeling: A review, *Practical Aspects of Computational Chemistry* , pp. 87–135.

Jacob, C., von Mammen, S., Davison, T., Sarraf-Shirazi, A., Sarpe, V., Esmaeili, A., Phillips, D., Yazdanbod, I., Novakowski, S., Steil, S., Gingras, C., Jamniczky, H., Hallgrimsson, B. and Wright, B. (2012). *Advances in Intelligent Modelling and Simulation: Artificial Intelligence-based Models and Techniques in Scalable Computing*, *Studies in Computational Intelligence*, Vol. 422, chap. LINDSAY Virtual Human: Multi-scale, Agent-based, and Interactive (Springer Verlag), pp. 327–349.

Kiefer, J. (1977). Conditional confidence statements and confidence estimators, *Journal of the American Statistical Association* **72**, 360, pp. 789–808.

Kiefhaber, R., Anders, G., Siefert, F., Ungerer, T. and Reif, W. (2012). Confidence as a means to assess the accuracy of trust values, in *Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, TRUSTCOM '12 (IEEE Computer Society, Washington, DC, USA), ISBN 978-0-7695-4745-9, pp. 690–697.

Larsson, T. and Akenine-Möller, T. (2001). Collision detection for continuously deforming bodies, in *Proc. Eurographics 2001*, pp. 325–333.

Lipson, H. and Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms, *Nature* **406**, pp. 974–978.

Louis, T. A. and Zeger, S. L. (2009). Effective communication of standard errors and confidence intervals, *Biostatistics* **10**, 1, p. 1.

Möller, T., Haines, E. and Hoffman, N. (2008). *Real-time rendering* (AK Peters Ltd.).

Predd, J. B., Kulkarni, S. and Poor, H. V. (2006). Distributed learning in wireless sensor networks, *Signal Processing Magazine, IEEE* **23**, 4, pp. 56–69.

Prothmann, H., Tomforde, S., Branke, J., Hhner, J., Mller-Schloer, C. and Schmeck, H. (2011). Organic traffic control, in C. Mller-Schloer, H. Schmeck and T. Ungerer (eds.), *Organic Computing A Paradigm Shift for Complex Systems*, *Autonomic Systems*, Vol. 1 (Springer Basel), ISBN 978-3-0348-0129-4, pp. 431–446.

Ramchurn, S., Jennings, N., Sierra, C. and Godo, L. (2004). Devising a trust model for multi-agent interactions using confidence and reputation, *Applied Artificial Intelligence* **18**, 9, pp. 833–852.

Rasmussen, S., Baas, N. A., Mayer, B., Nilsson, M. and Olesen, M. W. (2001). Ansatz for dynamical hierarchies, *Artificial Life* **7**, 4, pp. 329–353.

Sacerdoti, E. D. (1974). Planning in a hierarchy of abstraction spaces, *Artificial*

intelligence **5**, 2, pp. 115–135.

Sarraf Shirazi, A., Davison, T., von Mammen, S., Denzinger, J. and Jacob, C. (2013 (submitted)). Adaptive agent abstractions to speed up spatial agent-based simulations, *Simulation Modelling Practice and Theory* .

Sarraf Shirazi, A., von Mammen, S. and Jacob, C. (2010). Adaptive modularization of the mapk signaling pathway using the multiagent paradigm, in *Parallel Problem Solving from Nature – PPSN XI*, *Lecture Notes in Computer Science*, Vol. 6239 (Springer Verlag, Krakow, Poland), pp. 401–410.

Sarraf Shirazi, A., von Mammen, S. and Jacob, C. (2011a). Hierarchical self-organized learning in agent-based modeling of the mapk signaling pathway, in *CEC 2011, IEEE Congress on Evolutionary Computation* (IEEE Press, New Orleans, Louisiana), pp. 2245–2251.

Sarraf Shirazi, A., von Mammen, S. and Jacob, C. (2012 (in press)). Abstraction of agent interaction processes: Towards large-scale multi-agent models, *Simulation: Transactions of the Society for Modeling and Simulation International* .

Sarraf Shirazi, A., von Mammen, S., Yazdanbod, I. and Jacob, C. (2011b). Self-organized learning of collective behaviours in agent-based simulations, in H. S. et al. (ed.), *ICCS 2011: 8th International Conference on Complex Systems* (NECSI Knowledge Press, Boston, USA), pp. 543–555.

Schmidt, M. D. and Lipson, H. (2006). Actively probing and modeling users in interactive coevolution, in *GECCO '06: Proceedings of the 8th annual conference on Genetic and evolutionary computation* (ACM Press, New York, NY, USA), ISBN 1-59593-186-4, pp. 385–386, doi:http://doi.acm.org/10.1145/1143997.1144068.

Steghöfer, J.-P., Behrmann, P., Anders, G., Siefert, F. and Reif, W. (2013). Hispada: Self-organising hierarchies for large-scale multi-agent systems, in *ICAS 2013, The Ninth International Conference on Autonomic and Autonomous Systems*, pp. 71–76.

von Mammen, S. and Jacob, C. (2013). Into3d: Interaction-oriented 3d modelling and simulation, *Challenges: Special Issue on Challenges of Interface and Interaction Design* , p. 21.

von Mammen, S., Sarraf Shirazi, A., Sarpe, V. and Jacob, C. (2012). Optimization of swarm-based simulations, *ISRN Artificial Intelligence* **2012**, Article ID 365791, pp. 1–13.

von Mammen, S., Steghöfer, J.-P., Denzinger, J. and Jacob, C. (2011). Self-organized middle-out abstraction, in C. Bettstetter and C. Gershenson (eds.), *Self-Organizing Systems*, *Lecture Notes in Computer Science*, Vol. 6557 (Springer Verlag, Karslruhe, Germany), pp. 26–31.

Wang, Q., Megalooikonomou, V. and Faloutsos, C. (2010). Time series analysis with multiple resolutions, *Information Systems* **35**, 1, pp. 56 – 74, doi:DOI: 10.1016/j.is.2009.03.006, URL `http://www.sciencedirect.com/science/article/B6V0G-4W0R0PN-1/2/edba7b3395db108f1d76e8458c5399ce`.

Wilson, S. W. (1995). Classifier fitness based on accuracy, *Evolutionary Computation* **3**, 2, pp. 149–175.