Verification of Complex Adaptive Systems

Working Group Report - Dagstuhl Seminar 14512

Luca Bortolussi, Giacomo Cabri, Giovanna Di Marzo Serugendo, Vashti Galpin, Jane Hillston, Roberta Lanciani, Mieke Massink, Mirco Tribastone, Danny Weyns

Verification is the process of assessing how well a system meets a specification or requirement. A variety of approaches have appeared in the literature, ranging from model checking to static analysis of source code and theorem proving. In this working group, we primarily focused on verification based on model checking [2, 13], in which a state-based description of the system is assessed with respect to a property expressed in an appropriate specification language, like a temporal logic. In particular, we considered the challenges that arise in the model checking of Complex Adaptive Systems (CAS), which are systems comprised of a large number of heterogeneous agents which, interacting together, produce a complex array of collective behaviours.

In our working group in Dagstuhl, we first identified the major issues and more interesting challenges that arise in the process of verification of CAS. Afterwards, we divided in two subgroups to discuss in detail specific topics related to this general framework. In particular, we chose to investigate: (1) the quantitative or stochastic model checking in the presence of uncertainty, and (2) the specifications and logics which capture the spatial arrangements of systems, characterising the impact of those arrangements on collective behaviour. A brief account of each subgroup is given below.

1 Introduction

In our discussion, we first identified the numerous important issues and challenges that arise when we want to verify a CAS. In the following, we outline just a few of them.

1.0.1 Adaptation and verification

Adaptation is itself an interesting phenomenon which could be subjected to verification. In particular, we considered the issue of quantifying how adaptive the system is, and we identified a number of different measures that can be used to validate the adaptation of a CAS. For example:

Speed of adaptation - Once an adaptation is initiated, how rapidly does the system enter a stable behaviour? In this requirement the stable behaviour is not necessarily a single state, but could be a new stable equilibrium with a given probability distribution over a set of states.

Robustness - How often does the system adapt? Is there a danger of “thrashing”, meaning that the system alternates between different adaptations, achieving one, and shortly after pursuing another?

Effectiveness of adaptation - How closely does a system satisfy the revised property or goal after an adaptation?

1.1 Verifying global properties based on local behaviours

In many cases, the properties that are of interest to both system developers and system users are global requirements related to emergent behaviours of CAS. But the populations of CAS are comprised of a large number of single entities, whose local behaviour is usually more accessible and intuitive than the collective description of the system (due to the way CAS are implemented). Hence, there is a need for compositional approaches that are able to validate global properties of CAS, building on the verification of local requirements related to single agents or group of individuals. First promising results in this respect were achieved in [6] in the context of quantitative model checking of population models.

1.2 Verification in the presence of uncertainty

A characteristic feature of CAS is the uncertainty. For example, the structure of part of the system may be totally unknown or unknown at a particular instant in time. Moreover, the goals and objectives of a single agent may be hidden from the others, possibly due to an ongoing adaptation process. At a finer level of detail, the rates or probabilities that govern the dynamic behaviour of the system may be unknown, or changing in undefined ways, meaning that model of the CAS could be underspecified.

1.3 Scalability

Many verification techniques rely upon explicit representation of the state space of the model. In a CAS this is challenging in two respects. Firstly, not all possible states may be known due to future adaptation, as discussed above. Secondly, even if the "complete" state space is known or can be anticipated, the model will typically include too many states to be represented explicitly. Alternatives such as statistical model checking avoid constructing the whole state space at once, but then become computationally very expensive due to the sampling approach that must be adopted, necessitating many simulation runs before a verification can be approximated. An alternative is to use techniques based on fluid or mean field representation of the discrete state space [20, 4, 5, 6], but these approaches are still in their infancy.

1.4 Openness

Openness is an inherent property of CAS, as agents may join or leave the system throughout its lifetime. This poses severe challenges for state-based modelling techniques, particularly if there is the possibility that the population of the system grows unboundedly. In these scenarios, care is needed in phrasing the properties to be satisfied by the system. For example, it may be more appropriate to express goals in terms of proportions of agents rather than absolute numbers.

1.5 Quantified verification as a driver for adaptation

When the models contain quantitative information about the system, such as information about the timing and likelihood of events, it is possible to assess a system against a property not just in terms of boolean satisfaction but in a more quantified way. This can be viewed as measuring the degree to which a system satisfies a property, or the distance that a system is from satisfying the property. When this form of quantification is added to the verification process it is possible to see how verification can become a driver for adaptation. As the system undergoes adaptation, its progress towards a goal can be explicitly measured.

In recent years several authors have considered quantitative satisfaction of properties. In this framework, when a system is assessed against a property the result is a measure of distance indicating how close the system comes to satisfying the property [23, 21, 18]. For example, if the property is satisfied then the distance is zero. In this framework a system which fails to satisfy a property at distance 0.2 may be considered preferable to a system which fails to satisfy the property at distance 0.8. This approach has been used to conduct sensitivity analysis of model parameters with respect to desirable properties [24] and to seek parameters that bring a model closest to property satisfaction [25, 1]. This latter approach could be deployed to drive adaptation through verification.

1.6 Spatial aspects

In many CAS the location of agents, and bounded ranges of communication are an important factor in the design and realisation of the system. Thus it is essential that location and movement are treated as primitives in both modelling and verification techniques developed to support CAS. Often in existing techniques, if handled at all, space is treated only logically, and the relationships between locations are qualitative rather than quantitative. Thus a model may capture that locations are in some sense "adjacent" but not the actual distance between them. However, if agents are operating, for example, in a wireless sensor network, the actual distance between them will determine whether or not they are able to communicate, or the energy cost of doing so. Incorporating detailed spatial information means that model checking must consider spatio-temporal properties, a further level of challenge.

2 Motivation

To motivate our discussions in the working group we considered possible applications, where some of the discussed issues would have practical relevance. In particular, we identified the following:

Global adaptation: In this application, the adaptation takes place in the environment, while the agents operating within the system keep the same behaviour. An example of this would be a smart grid, where differential pricing is used to stimulate a change in the behaviour of the end users. In this scenario, a collective change in the dynamics of the system is obtained by acting on the environment (lowering the price of the energy during less busy period of the day), while the agents (the end users) keep the same goals and objectives (to buy energy at the cheapest possible price).

Agent adaptation: In these scenarios the agents change their behaviour based on local information, generating an effect on the collective behaviour of the system. An example of this is a peer-to-peer file sharing systems such as BitTorrent [14, 22]. In this application, the end users locally adapt to improve their own quality of service, while the environment, the BitTorrent protocol, remains unchanged. Moreover, the choices made by the single user affect its rate of uploading content to the network, thus altering the behaviour of the whole network.

3 Subgroup I: Uncertainty in CAS

When there is uncertainty in the behaviour of the system under consideration it makes the task of verifying a system even more challenging. Unfortunately in CAS, the inherent adaptability means that uncertainty is also inherent in the system. Broadly speaking, we identified two distinct approaches:

- Offline verification, before the system is deployed, tries to anticipate the possible range of behaviours that can be encountered.
- Online verification, conducted while the system is running, reflects the observed behaviour.

We anticipate that in many CAS both approaches may be needed.

3.1 Offline verification

In this case we might assume that a model is available which aims to capture all the possible behaviours, but that some aspect of the model, in particular the parameters corresponding to any given time or mode of behaviour, are unknown. To some extent this is the aim of probabilistic models which treat the parameters of models as random variations, with a defined distribution function which gives an estimate of the range of possible values and the likelihood of each. Such approaches have long been used in performance and dependability modelling to abstract away from concrete instances of behaviour and capture instead a range of possible behaviours in a statistically meaningful way. However, the uncertainty that we are considering here means that even the random variables characterising the range of behaviours may be unknown, or unknowable. This arises because we are often interested in designing systems satisfying emergent properties. Uncertainty can emerge in many ways; probably one of the simplest ways is to consider models that are structurally defined, but which can have unspecified parameter values p, possibly belonging to a bounded set P. Furthermore, we assume that we are dealing with stochastic models, so that behaviours are satisfied with a certain probability, rather than always or never.

The question of how we can verify properties under such uncertainty is a difficult one. One approach is to compute the probability of satisfaction of a certain property ϕ (for instance, encoded as a linear temporal logic formula) as a function of the parameter values $p \in P$. Let us denote this satisfaction function by $f(p)$. An exhaustive numerical computation (or bounding) of $f(p)$ for $p \in P$ is infeasible even for simple models. Recently, a statistical method was proposed by Bortolussi *et al.* [7], leveraging machine learning ideas to characterise statistically the function f. This approach can be complemented with that of [3], where the authors use a similar statistical scheme to evaluate a robustness measure associated with a property ϕ, and use it for system design purposes.

Applying these methods to CAS, however, still remains challenging. On one side, the size of such systems is so large that even fast statistical approaches can fail to provide an answer in a computationally acceptable time. In this respect, decomposition of the system into modules, seems an interesting direction for future work. The challenging problems here are how to identify such modules, and how to combine verification results of modules into a verification procedure (possibly providing bounds on the actual probabilities/robustness scores).

Another challenging problem is how to generalise the statistical approaches of [3, 7] to the case in which parameters are not only taking an unspecified value in a given set P, but they can also change over time (e.g. the arrival rate of customers at a bike or car sharing station will vary according to the time of day). This is a step towards verification of stochastic models of open CAS.

Finally, CAS are often subject to structural uncertainty, not only to parametrical one. In other words, the complete structure of the model itself may not be known. This seems to bring even more challenges to the analysis, particularly because structural changes of a model often result in discontinuous behaviour, which makes it much harder to exploit the statistical tools used in [3, 7] and similar work. Promising work on evolving models has recently been developed by Ghezzi *et al.* [17]. Whilst this was developed considering a specification that evolves as a system develops, it nevertheless has interesting ideas that could be applicable to the problem considered here.

3.2 Online verification

The importance of runtime verification has already been recognised by the software engineering community. In the simplest cases this might involve monitoring a system with respect to a formal specification of acceptable behaviour, such as a finite state machine, and raising an alarm if the observed behaviour deviates from the specification. However, systems such as CAS where the operating environment, the user requirements and the system itself are all changing over time, cannot be dealt with in such simplistic approaches. Moreover, functional correctness is no longer considered enough; in [9], the authors argue that quantitative aspects of behaviour must also be verified at runtime for self-adaptive software. In [19], the authors emphasise the need for formalising the adaptation components themselves, which is important to provide guarantees of correctness of the adaptation behaviour and changes of the adaptation logic to handle changing goals. Results from a related Dagstuhl seminar on Assurances for Self-adaptive Systems coined the term *perpetual assurances* [27] as an enduring process where new evidence is provided by combining system-driven and human-driven activities to deal with the uncertainties that the system faces across its lifetime.

In online verification a global model of the system may no longer be available, and if it is it is likely to be too computationally expensive to be used to give answers within the required timescale for runtime decision making. Thus it becomes attractive to develop compositional approaches to verification that allow results from lower levels, i.e. properties relating to one aspect of behaviour or relating to local behaviours, to be composed, building a global view from a number of local views. This may be addressed in a hierarchy, for example, with local verification giving rise to assurances about regional behaviour that can then be composed to give some assertion about global properties. It is likely that increasing levels of abstraction will be needed as we progress up the hierarchy, especially taking efficiency and response time into account. This remains a research challenge; whilst it is clear that boolean algebra provides the basis for composing verification results of qualitative analysis which lead to true/false answers, dealing with quantitative results of verification is more complex.

Another interesting approach for future work will be to combine statistical approaches with verification. For example, monitoring and verification could be combined with machine learning techniques to “learn”

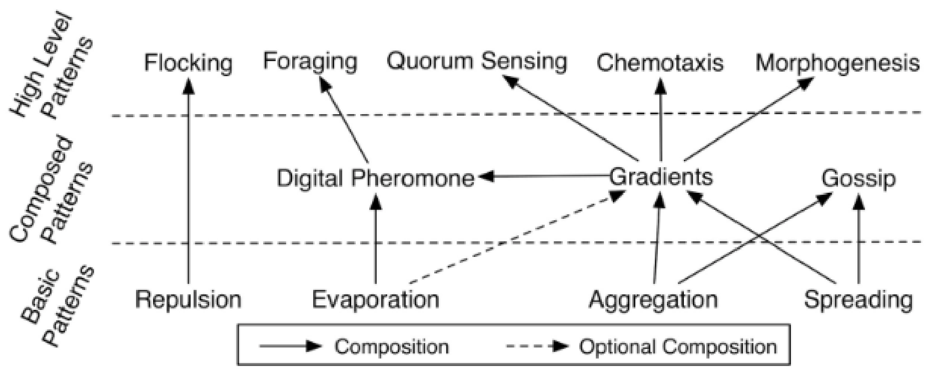


Figure 1: Patterns for self-organisation [16]

when a change in the system is likely to lead to an acceptable adaptation, or to guide adaptations in response to changes outside the system.

4 Subgroup II: Specification and verification of spatial self-organising patterns in CAS

In recent years a number of different frameworks have been proposed for the engineering of CAS. Here we particularly focus on the framework of self-organising patterns proposed in [16] and illustrated in Figure 1. Examples of these patterns include:

Spreading: a copy of the information (received or held by an agent) is sent to neighbours and propagated over the network from one node to another. Information spreads progressively over the system.

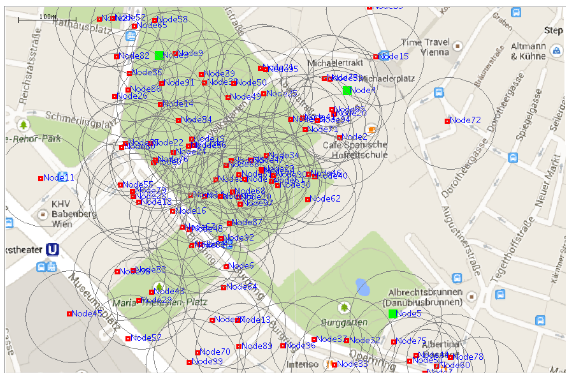
Aggregation: information is distributively processed in order to reduce the amount of information and to obtain meaningful information. Aggregation consists of locally applying a fusion operator to synthesise macro information (filtering, merging, aggregating, or transforming).

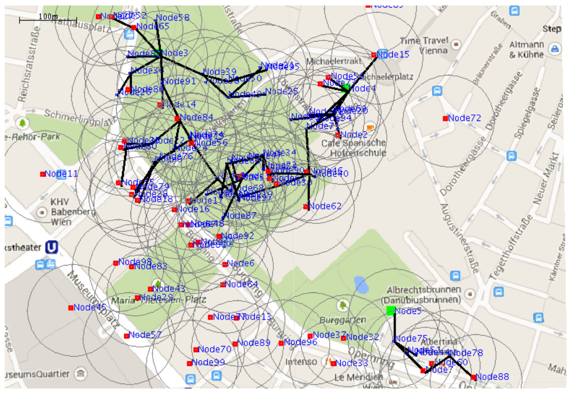
Gossip: in large-scale systems, agents need to reach an agreement, shared among all agents, with only local perception and in a decentralised way. Information spreads to neighbours, where it is aggregated with local information. Aggregates are spread further and their value progressively reaches the agreement.

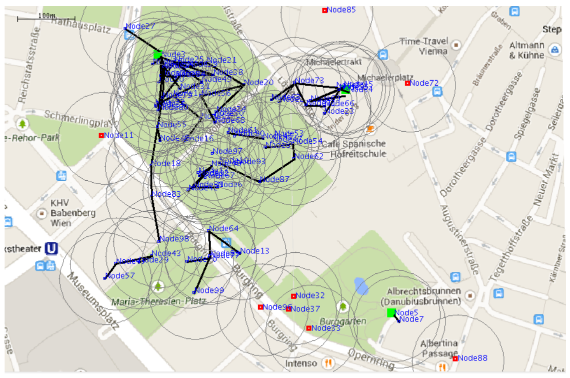
Gradient: information spreads from the location where it is initially deposited and aggregates when it meets other information. During spreading, additional information about the sender's distance and direction is provided: either through a distance value (incremented or decremented); or by modifying the information to represent its concentration (lower concentration when information is further away).

These patterns may be combined to design a system with collective adaptive agents who achieve a desired high-level outcome. For example, consider a scenario of emergency egress in a city. People are assumed to have handheld devices on which they can receive real-time information about the directions to follow to the nearest exit. This information is propagated via their neighbours. However, these devices have only a limited radius for local communication. The idea is therefore to create a dynamic ad-hoc network that aims to eventually reach everyone and provide the required information using dynamic gradients. Figure 2 shows four snapshots of a particular evolution of the system in the initial state and some later times.

When we consider verification there are a number of properties that are of interest in this case study. We mention just a few examples.


(a) Before Evacuation


(b) Emergency Egress propagate gradients


(c) Nodes follow Gradients' information

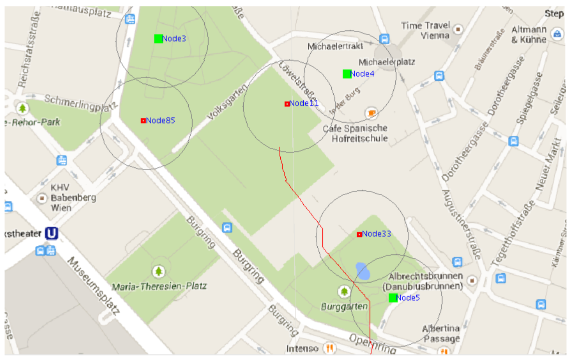

(d) End of Evacuation

Figure 2: Four snapshots of the emergency egress scenario: The emergency exits are the green boxes, the people that have not been reached yet are indicated by red boxes and the circles around people show the radius within which they can get in touch with their neighbours. After some time the dark lines indicate the gradient structure.

- As the mechanism of the communication is dependent on the gradient, it is important to assess its functional properties such as whether all nodes will be reachable.
- By chemotaxis, the agent should follow the gradient through the shortest path. Verification can assess whether all agents will reach the source of the gradient.
- We can also consider quantitative aspects of the induced behaviour such as the speed with which information is spread, the bandwidth consumed and the probability of reaching a certain fraction of people within a certain time.
- Spatio-temporal properties are also of concern such as whether at some future time all reachable nodes receive the gradient information (the spatial dispersion of information at a given time), or conversely at some given location all reachable nodes will receive the gradient information within a given time.
- Invariant properties may be important such as ensuring that the shortest paths are built and that the generated gradient structure does not have loops.

When an application is developed based on such patterns, the details of implementation are often delegated to an agent or service that implements the pattern in question. Therefore it is important that we have the means to independently verify the properties of the supplied patterns (local behaviours) as well

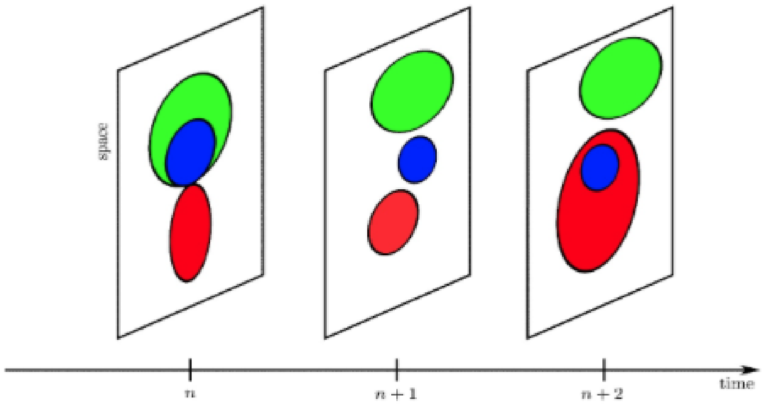


Figure 3: Schematic view of the evolution of a temporal path.

as checking the emergent properties at the global level. In particular, it would be hugely beneficial to be able to develop mechanisms for compositional verification to use the verified properties of each pattern to derive the properties of the higher-level patterns, and ultimately properties of the applications build using those patterns. For example, considering the high-level patterns in Figure 1, is it possible to derive the correctness of a flocking pattern in a straightforward manner given the correctness of the repulsion pattern used to implement it?

4.1 Spatio-temporal verification via model-checking

Properties of different patterns at the collective level emerge from the coordination at the local level. Considering the example of dynamic gradients used to guide people to the nearest exits in an emergency situation such as that shown in Figure 2 [15], various global properties of the collective mechanism are of interest. For example, as already mentioned, one might like to be sure that all people involved at any time do receive gradient updates on directions within a given time.

Some such properties are spatio-temporal in nature and spatial [12, 10] and spatio-temporal model-checking [11] could be one of the techniques to be considered to automatically verify such properties. Spatio-temporal model-checking is a technique that requires a spatio-temporal model, on the one hand, and a spatio-temporal property, on the other. One way in which the model can be conceived is to consider it as a composition of a Kripke structure (S, T) to model the temporal evolution of the system and a spatial model for each state of the Kripke structure that reflects the spatial situation at the particular time. The latter can also be imagined as a "snapshot" of the spatial situation in a state of the Kripke structure. The spatial model can be conveniently chosen to be a closure space (X, C), where X is a set of points, and C a closure operator which has its origin in topological spatial logics [26]. Such a choice covers a wide range of specific choices for the spatial structure, encompassing graphs, regular grids and also images similar to those shown in Figure 2.

Spatio-temporal properties address both the evolution in time and the spatial properties of a point in the model, i.e. a point (node) in the space at a particular state in the Kripke structure. In each possible world there is a different valuation of atomic propositions, inducing a different "snapshot" of the spatial situation which "evolves" over time. This is made clear along a temporal path. A path in the Kripke structure denotes a sequence of digital pictures indexed by instants of time. This is illustrated more schematically in the Figure 3, showing three different states of the Kripke structure at time step n, $n+1$ and $n+2$.

Spatio-temporal operators in STLCS (Spatio-Temporal Logic for Closure Spaces) [10] feature the usual Boolean operators (negation, disjunction etc), the CTL path quantifiers A ("for all paths"), E ("exists a path"), which must be followed by path-specific temporal operators XF ("in the next step"), $F1\ U\ F2$, ("eventually $F2$ holds, but until then $F1$ must hold"), where $F, F1$ and $F2$ are STLCS formulas, and the spatial operators closure C and spatial until $F1\ S\ F2$ ("$F1$ surrounded by $F2$"). The two derived temporal operators G ("always") and F ("eventually") are also very useful.

Let us proceed with a few simple examples.

- Consider the STLCS formula $EG(green\ S\ blue)$. This formula is satisfied at a point x in the graph, associated with the initial state $s0$, if there exists a (possible) evolution of the system, starting from $s0$, in which point x is *always*, i.e. in every state in the path, *green* and surrounded by *blue*. The prototype spatio-temporal model-checker described in [10] will return (or colour) all the points x that satisfy the formula.
- A further, more complicated, nested example is the STLCS formula

$$EF(green\, S\ (AX\, blue)).$$

  This formula is satisfied at a point x in the graph associated with the initial state $s0$, if there is a (possible) evolution of the system, starting from $s0$, in which point x is eventually *green* and surrounded by points y that, for every possible evolution of the system from then on, will be *blue* in the next step.
- A simple example concerning the emergency egress case is the formula $AF(!red\ S\ \mathsf{false})$, where ! denotes negation. This formula is satisfied at a point x in the initial state if all possible evolutions of the system eventually reach a state in which there are no red points. Recall that red points correspond to nodes representing people who did not receive the directions to the nearest exit. So when this formula is satisfied it means that there is a point in time at which all people are being updated by the system. The particular expression $!red\ S\ \mathsf{false}$ is satisfied if none of the pixels are red because the surround operator is a spatial variant of a weak until operator.

This is an area of on-going work and there are a number of open issues for this line of research.

- Are the STLCS spatio-temporal operators sufficient to express the spatio-temporal properties that are relevant to the self-organising patterns used to design and implement CAS?
- If not, which other operators would be needed? Here we can think of operators that address performance aspects, for example to express that the probability that 90% of the people in the emergency egress example have been reached within a certain time-bound T, or more collective aspects such as that a set of points satisfies a certain property.
- What would be the right set of basic operators that on the one hand provide satisfactory expressiveness, or at least cover an interesting class of properties, and on the other hand are such that efficient model-checking algorithms can be found to verify them?
- Which derived operators and property templates are convenient to facilitate formulation of relevant properties in this context?

Much interesting and challenging work remains to find answers to these questions. Furthermore, there are other proposals that consider spatial and spatial-temporal logics. As an example, we would like to mention the spatial signal temporal logic [8]. This is a linear logic for the specification of behavioural properties of continuous signals which has recently been extended with some spatial operators. This logic has been applied in the domain of epidemiology to analyse the spread of a virus.

5 Concluding remarks

An important aspect of the engineering of CAS is providing evidence that the system requirements are satisfied, despite the uncertainty that may affect the system, its goals, and its environment. In this working group, we primarily focussed on verification of CAS to assess how well a system meets its specification or requirements. The specifics and characteristics of CAS poses several challenges to the problem of verification, including:

- How to express emergent properties of CAS and verify them?
- How to provide evidence in the face of uncertainty, in particular structural uncertainty, where the complete structure of the model of the system may not be known?
- How to enable runtime verification for CAS for which a global model is not available?
- How to enable compositional verification that uses verified properties of patterns at lower levels to derive the properties of higher-level patterns and ultimately properties of CAS built using those patterns?
- How to express spatio-temporal properties relevant for self-organising systems used to design and implement CAS?
- How to blend offline with online verification to provide the evidence that the system requirements are satisfied during the entire lifetime of CAS?

Whilst we enjoyed and benefited from a number of stimulating discussions around the topic of verification of CAS during the Dagstuhl seminar 14512, the time available did not allow us to make any significant developments beyond deepening our understanding and mutual appreciation. Nevertheless, the discussion helped us to hone our ideas and identify a number of exciting topics for future research, several of which are now being actively pursued by members of the working group.

References

[1] Eugene Asarin, Alexandre Donzé, Oded Maler, and Dejan Nickovic. Parametric identification of temporal properties. In *Runtime Verification - Second International Conference, RV 2011, San Francisco, CA, USA, September 27-30, 2011, Revised Selected Papers*, volume 7186 of *Lecture Notes in Computer Science*, pages 147–160. Springer, 2011.

[2] C. Baier and J.P. Katoen. *Principles of Model Checking.* MIT press, 2008.

[3] Ezio Bartocci, Luca Bortolussi, Laura Nenzi, and Guido Sanguinetti. On the robustness of temporal properties for stochastic models. In *Proceedings Second International Workshop on Hybrid Systems and Biology, HSB 2013, Taormina, Italy, 2nd September 2013.*, volume 125 of *EPTCS*, pages 3–19, 2013.

[4] Luca Bortolussi and Jane Hillston. Fluid model checking. In *CONCUR 2012 - Concurrency Theory - 23rd International Conference, CONCUR 2012, Newcastle upon Tyne, UK, September 4-7, 2012. Proceedings*, volume 7454 of *Lecture Notes in Computer Science*, pages 333–347. Springer, 2012.

[5] Luca Bortolussi and Roberta Lanciani. Model checking Markov population models by central limit approximation. In *Quantitative Evaluation of Systems - 10th International Conference, QEST 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8054 of *Lecture Notes in Computer Science*, pages 123–138. Springer, 2013.

[6] Luca Bortolussi and Roberta Lanciani. Stochastic approximation of global reachability probabilities of Markov population models. In *Computer Performance Engineering - 11th European Workshop, EPEW Florence, Italy*, volume 8721 of *Lecture Notes in Computer Science*, pages 224–239. Springer, 2014.

[7] Luca Bortolussi, Dimitrios Millios, and Guido Sanguinetti. Smoothed model checking for uncertain continuous time Markov chains. *CoRR*, abs/1402.1450, 2014.

[8] Luca Bortolussi and Laura Nenzi. Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic. In *VALUETOOLS 2014*, In Press.

[9] Radu Calinescu, Carlo Ghezzi, Marta Z. Kwiatkowska, and Raffaela Mirandola. Self-adaptive software needs quantitative verification at runtime. *Commun. ACM*, 55(9):69–77, 2012.

[10] Vincenzo Ciancia, Stephen Gilmore, Diego Latella, Michele Loreti, and Mieke Massink. Data verification for collective adaptive systems: spatial model-checking of vehicle location data. In *2nd FoCAS Workshop on Fundamentals of Collective Systems*, IEEE Eight International Conference on Self-Adaptive and Self-Organizing Systems, page to appear. IEEE Computer Society, 2014.

[11] Vincenzo Ciancia, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. A spatio-temporal model-checker, `http://blog.inf.ed.ac.uk/quanticol/technical-reports/`. Technical report, The QUANTICOL project, 2014.

[12] Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Specifying and Verifying Properties of Space. In Springer, editor, *The 8th IFIP International Conference on Theoretical Computer Science, TCS 2014, Track B*, volume 8705 of *Lecture Notes in Computer Science*, pages 222–235, 2014.

[13] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking.* MIT press, 1999.

[14] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[15] Jose Luis Fernandez-Marquez and Giovanna Di Marzo Serugendo. Assessment of robustness of spatial structures in crowd steering scenarios. Technical Report SAPERE TR.WP2.2013.07, SAPERE Project, 2013.

[16] Jose Luis Fernandez-Marquez and Giovanna Di Marzo Serugendo. From self-organizing mechanisms to design patterns to engineering self-organizing applications. In *7th IEEE International Conference on Self-Adaptive and Self-Organizing Systems, SASO 2013, Philadelphia, PA, USA, September 9-13, 2013*, pages 267–268. IEEE Computer Society, 2013.

[17] Carlo Ghezzi, Claudio Menghi, Amir Molzam Sharifloo, and Paola Spoletini. On requirement verification for evolving statecharts specifications. *Requir. Eng.*, 19(3):231–255, 2014.

[18] Thomas A. Henzinger. Quantitative reactive modeling and verification. *Computer Science - R&D*, 28(4):331–344, 2013.

[19] M. Usman Iftikhar and Danny Weyns. Activforms: Active formal models for self-adaptation. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS 2014, pages 125–134, New York, NY, USA, 2014. ACM.

[20] Diego Latella, Michele Loreti, and Mieke Massink. On-the-fly fast mean-field model-checking. In *Trustworthy Global Computing - 8th International Symposium, TGC 2013, Buenos Aires, Argentina, August 30-31, 2013, Revised Selected Papers*, volume 8358 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2013.

[21] Radu Mardare, Luca Cardelli, and Kim G. Larsen. Continuous Markovian logics - axiomatization and quantified metatheory. *Logical Methods in Computer Science*, 8(4), 2012.

[22] Dongyu Qiu and Rayadurgam Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 367–378. ACM, 2004.

[23] Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. On a continuous degree of satisfaction of temporal logic formulae with applications to systems biology. In *Computational Methods in Systems Biology, 6th International Conference, CMSB 2008, Rostock, Germany, October 12-15, 2008. Proceedings*, volume 5307 of *Lecture Notes in Computer Science*, pages 251–268. Springer, 2008.

[24] Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. A general computational method for robustness analysis with applications to synthetic gene networks. *Bioinformatics*, 25(12), 2009.

[25] Aurélien Rizk, Grégory Batt, François Fages, and Sylvain Soliman. Continuous valuations of temporal logic specifications with applications to parameter optimization and robustness measures. *Theor. Comput. Sci.*, 412(26):2827–2839, 2011.

[26] Johan van Benthem and Guram Bezhanishvili. Modal logics of space. In Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors, *Handbook of Spatial Logics*, pages 217–298. Springer, 2007.

[27] Danny Weyns, Nelly Bencomo, Radu Calinescu, Javier Camara, Carlo Ghezzi, Vincenzo Grassi, Lars Grunske, Poala Inverardi, Jean-Marc Jezequel, Sam Malek, Raffaela Mirandola, Marco Mori, and Giordano Tamburrelli. Perpetual assurances in self-adaptive systems. In *Assurances for Self-Adaptive Systems, Dagstuhl Seminar 13511*, 2014.