Compositional transient reachability analysis for agent-based simulations

Benjamin Herd [*], Simon Miles [*], Peter McBurney [*], Michael Luck [*]

[*] *Department of Informatics*
King's College London
London, United Kingdom

Abstract. *Even though agent-based simulations have been applied successfully to various real-world projects, there is still much reluctance to accept them to the same degree as more traditional, equation-based techniques. Their inherent complexity due to nonlinear dynamics, a high level of heterogeneity and a vast parameter and state space are just some of the properties that make them difficult to understand, to verify and to validate. We analyse the particular characteristics of agent-based simulations and give recommendations for a model checking-based verification process. Furthermore, we sketch a preliminary verification approach which is based on an iterative, compositional and non-exhaustive model construction process and transient reachability analysis.*

Keywords: *Agent-based simulation, verification, reachability analysis, model checking, transient analysis*

1. Introduction

During the last two decades, the popularity of agent-based computing and the application of agent-based systems to the solution of real-world problems has increased significantly [LMP04]. With their distributed nature, their complex dynamics and the adaptive and autonomous characteristics of their constituents, they provide a powerful tool for the creation of systems which are capable of solving complex problems efficiently [BML$^+$06]. A subarea of agent-based systems, *agent-based simulation (ABS)* or *agent-based modelling (ABM)*, deals with the simulation of real-world phenomena using a multiagent approach [Bon02]. Especially for the simulation of complex adaptive systems like human societies, markets and biological networks, ABS has proven to be a powerful alternative and in many cases even superior to more classical approaches like econometrics, game theory and system dynamics [PSR98]. Despite the efforts that have been made during the last few years, however, ABSs are still often considered black boxes whose dynamics are hard to understand. The lack of common definitions, formalisms and rigorous analysis techniques as well as the empirical nature of the agent-based modelling and simulation process contributes to the scepticism with which ABS is often perceived by advocates of more established techniques.

Similar to other software development efforts, aspects of robustness and correctness also play a central role in agent-based modelling and questions of quality assurance become increasingly important [Mar07, MMK07]. In this context, it is important to distinguish between *verification* and *validation*. Whereas the former is targeted towards a system's correctness with respect to its specification (i.e. correct implementation), the latter ensures a sufficient level of accuracy with which it it consistent with the intended application domain, e.g. the real-world phenomenon in an ABS [Sar08]. In this paper we focus on verification, i.e. the assessment of the expected behaviour of the simulation against the background of a given specification rather than its accuracy regarding the reproduction of a real process.

For the more general area of agent-based systems, verification has gained considerable attention throughout the last decade [BFW09, BFVW06]. However, due to the particular characteristics and requirements of ABSs, existing approaches in the area of general agent-based systems cannot be simply adapted to their simulation counterparts. This paper attempts to address this problem and considers verification, particularly *model checking*, against the background of ABSs. Our main contributions are:

1) We analyse the influence of ABS properties on the choice of an appropriate model checking methodology

2) We sketch a preliminary approach for the verification of transient reachability properties which is capable of analysing large-scale ABSs efficiently

The paper is organised as follows. In Section 2 we describe the motivation underlying our work and describe the necessity for appropriate verification techniques in the area of agent-based modelling. Section 3 gives an overview of existing work on verification of agent-based systems and simulations with a particular focus on model checking. Section 4 describes our observations about typical properties of ABSs and recommendations for a potential verification process. In Section 5 we describe our reachability analysis approach in further detail, particularly the calculation of transient probabilities. We conclude the paper with a summary of our work and ideas for future research.

2. Background

Since agents continue to become a widely accepted paradigm of modern information technology based on which an increasing number of complex real-world applications are built, questions of correctness and quality management become more and more important. In the agent area, most verification approaches that have been developed so far focus primarily on general multiagent systems (MAS) which comprise a small number of components. Even though a high number of agents is not a necessary requirement for ABSs, it is certainly a typical one. Furthermore, current MAS verification efforts are mostly targeted towards agent internals such as goals, plans and actions [LQR09, BFVW06]. Even though these aspects might also be important for ABSs, additional techniques for the verification of highly probabilistic and emergent systems are urgently required. Approaches that put a special emphasis on ABSs and take into account their particular characteristics are still missing.

Model checking [Cla97] is a special type of formal verification which uses a finite state model of the target system together with a specification of desired properties which is typically given in some temporal logic, e.g. LTL, CTL or CTL*. The verification of the system's correctness is then achieved by checking whether a given property holds in all possible execution paths. In the area of general multiagent systems, model checking has been investigated extensively [LQR09, BFVW06]. For systems that exhibit random behaviour, probabilistic variants of model checking have been proposed [HJ94, BdA95]. As an underlying formal model, they use a probabilistic finite state representation of the target system, typically a Markov chain. As opposed to conventional model checking, probabilistic model checking allows for the verification of both *qualitative* and *quantitative* temporal properties.

A central problem of all model checking efforts is *state space explosion*, i.e. the exponential growth of the target system's state space. Several techniques have been developed to address this problem, e.g. symbolic model checking, which avoids representing the state space explicitly [McM92]. Due to the much more compressed representation (e.g. using binary decision diagrams), symbolic model checking approaches are typically significantly more efficient than their explicit-state counterparts. With symbolic model checking it is possible to include hundreds of state variables into the verification process and check systems with up to 10^{120} states [CGJ$^+$01]. Other popular approaches that attempt to alleviate or circumvent combinatorial explosion are symmetry- and partial-order reduction [CEJS98, CGMP99], compositional verification [HQR98], SAT-based model checking [CBRZ01], abstraction [FLW06] and approximation [HLMP04].

Even though the advances in terms of state space reduction are impressive, the size of most real-world ABSs is still multiple orders of magnitude larger than what model checking algorithms are able to handle efficiently. This is particularly critical when global properties of a system need to be examined, i.e. when the correctness of the system cannot be simply reduced to the correctness of particular components. In agent-based modelling, the particular focus of interest is typically on global emergent properties rather than just on a single agent's behaviour or the interaction between a small group of agents. In order to assess the global correctness of an ABS, one needs to look at the evolution of the entire system over time. The central challenge in model checking, the efficient construction of the state space and the avoidance of exponential growth, is therefore crucial for the verification of large-scale ABSs. Current model checking approaches do not take into account the particular characteristics of ABSs. In this paper we attempt to address this problem and investigate how these characteristics can be successfully exploited in order to make model checking more efficient and allow for the verification of large-scale ABSs.

3. Related work

Despite the growing importance of ABS, dedicated verification techniques for their analysis are still few and far between. In recent years, probabilistic model checking has gained increasing importance for the verification of general multiagent systems (of which ABSs can be conceived as a special case). An interesting approach to verify the emergent behaviour of robot swarms using probabilistic model checking has been presented by Konur et al. [KDF10]. In order to tackle the combinatorial explosion of the state space, the authors exploit the high level of symmetry in the model and use *counter abstraction* [FLW06]. Instead of creating the parallel composition of the single agents' state machines, they represent the system with a single, system-level state machine. This global representation is similar to the individual state machines but contains an additional counter which determines how many individual agents are in the current state. In doing so, the authors manage to transform the originally exponential into a polynomial problem [1]. This is a significant improvement, however, since the resulting problem is still exponential in the number of agent states, the approach remains limited to small-scale systems. Ballarini et al. [BFW09] apply probabilistic model checking to a probabilistic variant of a negotiation game. They use PRISM [KNP02], a probabilistic model checker, to verify PCTL-based properties addressing two aspects of the system: i.) the value at which an agreement between two agents bargaining over a single resource is reached and ii.) the delay for reaching an agreement. In this scenario, the overall state space is small and therefore combinatorial explosion is not an issue. According to the authors, probabilistic verification provides an interesting alternative to analytical and simulation methods and can provide further insight into the system's behaviour.

1. To be precise: The resulting problem is polynomial in the number of agents and exponential in the number of agent states

Dekhtyar et al. [DDV08] assume that randomness in a multiagent system can be due to i.) uncertainty of communication channels and ii.) uncertainty of action. In their paper, they describe a method to translate a multiagent system into a finite-state Markov chain and analyse the complexity of probabilistic model checking of its dynamic properties. Apart from mentioning the exponential complexity of both state space creation and verification, however, the authors do not present any ways to circumvent this problem. The verification of epistemic properties has also been addressed against the background of probabilistic agent-based systems. Wan et al. [WBBH11] propose PCTLK, an epistemic, probabilistic branching-time logic which extends CTL with probabilistic and epistemic operators. In their paper, the focus of interest is rather on agent internals and thus complexity issues are not being addressed.

A different formal approach to the analysis and verification of agent-based simulations has been proposed by Izquierdo et al. [IIGS09]. The authors describe how simulations can be encoded into time-homogeneous Markov chains and analysed in terms of their transient and steady-state behaviour. In order to illustrate this approach, they use popular social simulation models such as Schelling's model of spatial segregation [Sch69], Sugarscape [EA96] or Axelrod's metanorms model [Axe86]. Since the main focus of the paper is on the usefulness of Markov chain analysis for the understanding of complex simulation models, the authors do not provide any state space reduction techniques in order to circumvent the combinatorial explosion. However, they describe ways of analysing the behaviour without having to represent the transition matrices. More specifically, they make assumptions about the nature of the state space and derive insights by pure reasoning, e.g. whether the system will eventually reach an absorbing state and stay there forever.

4. Observations of ABSs

An ABS is a special kind of software system that combines elements of both agent-based systems and computer simulations. Despite the variety of problem domains and models presented in literature, we found that most ABSs share some common characteristics which can be helpfully exploited in order to guide a model checking-based verification process. We are currently investigating these characteristics with the overall goal of integrating them into a formal, abstract ABS framework. In the following paragraphs, we will present our preliminary findings and describe some of the characteristics that we found most crucial. We will use them in order to give recommendations for a suitable verification process.

Randomness: In order to represent uncertainty in an agent's decision process as well as to introduce heterogeneity into the population, ABSs often exhibit a significant amount of randomness. As a consequence, questions about the temporal behaviour of the ABS do not only yield simple yes/no answers but also quantitative statements about the probability distribution of states. A verification technique needs to take into account the probabilistic nature of the underlying system and be able to answer both qualitative and quantitative questions.
$\Rightarrow$ ***A probabilistic variant of model checking should be used***

Emergence: One of the central features of ABSs that makes them a powerful tool for the simulation of real-world complex systems is their ability to produce emergent behaviour. ABSs can be viewed as comprising at least two different levels – the *micro level* where individual agents with their local perceptions and (inter)actions are situated, and the *macro level* where the emergent phenomena can be observed. The correlation between the two levels can be difficult to understand and, in order to determine causal relationships, comprehensive experiments need to be conducted. So as to address this problem and ensure that the global emergent behaviour of the ABS complies with the developer's intention, a verification technique should not only be able to prove the correctness of single components, but also of the system's global behaviour.
$\Rightarrow$ ***A verification technique should be able to verify global, system-wide temporal properties***

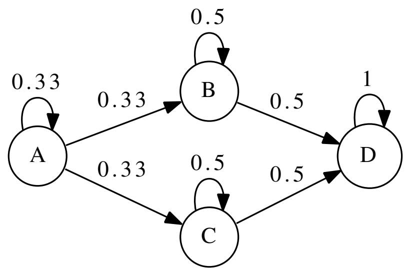
Discrete time structure: ABSs are typically based on a discrete time structure, i.e. their execution progresses on a tick-per-tick basis. Within each tick or time step, agents update their state (either synchronously or asynchronously). This suggests an incremental model construction process which, starting from an initial state, expands the

state space until the maximum number of time steps has been reached or a given property has been proved or disproved.
$\Rightarrow$ ***The formal model should be constructed in an incremental way***

Time-boundedness: ABSs are typically executed for a limited number of time steps only. A verification method can focus on the temporal behaviour during these time steps and ignore anything beyond. Conventional techniques do not involve any notion of time and attempt to construct the space exhaustively. For efficiency reasons we propose to focus on a partial state space which only covers the behaviour of the system up to the maximum time step.
$\Rightarrow$ ***The formal model should focus on the relevant fraction of the state space***

Fungibility: ABSs typically exhibit a high level of symmetry. Instead of designing each agent separately, variations are usually introduced through randomness in the agent's behaviour. As a consequence, ABSs are often highly homogeneous in nature and despite their considerable size, only few different agent types are actually represented in the population. The instances of a particular agent type are *fungible*, i.e. they can be exchanged without impacting the system's dynamics. This high level of symmetry can provide a useful basis for reducing the state space.
$\Rightarrow$ ***The state space can be reduced through appropriate symmetry reduction and abstraction techniques***

Modularity: An ABS can be seen as a highly distributed system that consists of a (potentially huge) number of loosely coupled, autonomous and interacting components. The interaction network between agents is often neither regular (e.g. a grid) nor completely random. Instead, it has been found that real-world networks like social, biological or computer networks often naturally divide into communities [New06]. It can be assumed that agents within communities or cliques are more likely to communicate with each other than with agents in different communities [Cou85]. For a verification approach, this modularity means that both model construction and verification can be achieved compositionally on a per-agent and per-community basis.
$\Rightarrow$ ***Model checking should be done in a compositional, bottom-up way***



		Current state			
		A	**B**	**C**	**D**
Next state	**A**	0.33	0.0	0.0	0.0
	B	0.33	0.5	0.0	0.0
	C	0.33	0.0	0.5	0.0
	D	0	0.5	0.5	1.0

Figure 1: Graph representation and adjacency matrix of a simple agent DTMC

On top of these particular characteristics lies the general complexity of ABSs. They can be large-scale systems which can easily comprise hundreds, thousands or even millions of constituents, each of which can be of arbitrary logical complexity itself. The exponential growth of the state space makes it difficult and in most cases even impossible to analyse and check all possible execution paths for incorrect behaviour – even for small artificial systems. One way to address this problem is to find suitable approximations to the optimal verification result. Since an ABS is typically not used in a safety-critical environment, an approximate but fast solution is in most cases preferable over a precise but extremely time-consuming one.

In the next section, we sketch a possible verification approach which follows the recommendations given above and allows for the verification of simple reachability properties over large-scale ABSs.

5. Reachability analysis

In the previous section we described some typical characteristics of ABSs and their implications on model checking. In this section we outline a possible verification approach which aims to exploit these characteristics. It follows the recommendations mentioned in the previous section such that it is **probabilistic**, **iterative**, **compositional** and it allows for the verification of **global, system-wide properties** of **large-scale ABSs** through simple reachability analysis. Furthermore it allows for **abstraction** and **approximate answers** through reduction operations that prune the state space on-the-fly using different criteria. In order to address the combinatorial explosion problem, we avoid construction of the entire system's state space. Instead, we propose an approach based on transient analysis of agent-specific probabilistic finite state representations which are expanded incrementally. On top of that we use a numeric technique to calculate the global behaviour of the system during the verification phase.

Before we give a more detailed description of the actual verification approach in Section 5.1, we need to introduce some basic concepts. We assume that the behaviour of a single agent can be represented as a probabilistic finite-state machine (FSM). Each state of the FSM represents a state of the agent, a transition between states represents an agent's probabilistic choice of action which leads to an update of its current state. We can assume that all agent-related information including its knowledge, goals, memories etc. can be compressed into its state. The FSM therefore satisfies the *Markov property* which states that all outgoing transitions depend on the current state only and not on the sequence of past states. In combination with the discrete time structure based on which an agent progresses, its FSM thus reduces to a *Discrete-time Markov Chain (DTMC)* which we will describe formally in Section 5.2. A population of agents can then be seen as a set of individual DTMCs – one for each agent.

A DTMC can be depicted as a *directed graph* where states act as vertices and there is an edge between two states s and s' if and only if the probability of moving from s to s' is greater than 0. Furthermore, a DTMC can also be described with an *adjacency* or *transition matrix*. An example using both representations is given in Figure 1. The DTMC comprises four states A, B, C and D and we assume that it represents an agent's evening planning. Therefore we assign the following semantics to the abstract states:

– State A: Agent is in the office

– State B: Agent is in the shop

– State C: Agent is in the pub

– State D: Agent is at home

Initially, our agent is in the office (state A). It has three possible choices: It could stay longer, since there is still much work to do. It

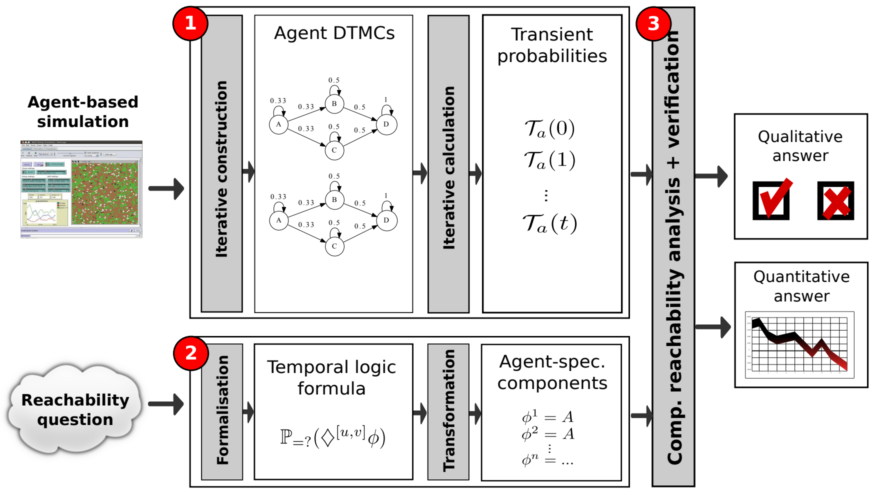


Figure 2: Schematic overview of the verification approach

could also go shopping (move to B) or it could join some friends in the pub for some drinks (move to C) . Our agent is rather undecisive and therefore its chances of staying in the office, going to the shop and going to the pub are equally distributed to $\frac{1}{3}$. Both in the pub and in the shop, the agent can either decide to stay or to return home (move to D) with a probability of $\frac{1}{2}$, respectively. After returning home, the agent will stay there with a probability of 1 and not leave the house anymore.

5.1. *Outline of the approach*

Our verification approach is depicted schematically in Figure 2. It consists of three basic steps which are marked by red circles in the diagram:

1) Iterative model construction and calculation of agent-specific transient probabilities

2) Transformation of a desired global state into its agent-specific components

3) Compositional reachability analysis and verification

The first step involves the state space construction and the calculation of transient probabilities *separately for each agent*. Here, we need to distinguish three different scenarios with increasing complexity:

– Scenario 1: Transition probabilities are known in advance

– Scenario 2: Transition probabilities are not known in advance

– Scenario 3: Agents are interacting

In the simplest case (scenario 1) we assume that the state spaces of the agents are known in advance, i.e. all states and transitions are initially available to the verification algorithm. In reality, this will rarely occur since it contradicts the adaptive and interactive principle of agents. Nevertheless, it serves well as an explanatory scenario to convey the central ideas. In this case, the transient probabilities can be obtained using simple vector-matrix calculations (see Section 5.3). In the second scenario we assume that agents are still independent from eacher but their transition probabilities are not known in advance. This might for example be the case for simple microsimulation models where there is adaptive behaviour (e.g. depending on some environmental conditions) but no interaction between the agents. In this case, the transient probabilities can be calculated using an iterative process (see Section 5.4). The third scenario takes into account interaction between agents by proposing an additional perception step (see Section 5.5).

The second step involves the transformation of an (informal) reachability question into a formula in temporal logic and the transformation of a global system state (whose reachability probability is to be determined) into its agent-specific components (the local states).

The third step comprises the actual verification through compositional reachability analysis. By multiplying the individual transient probabilities of the agent-specific local states, we determine the transient probability of the global state which can then be either used to prove or disprove the property (in the qualitative case) or to determine probability values or distributions (in the quantitative case). It is important to note that, by using a numerical approach to determine the global probabilities, we avoid to construct the entire system's state space and thus reduce the memory and computation requirements significantly. Nevertheless, the single agent's state spaces might still be subject to exponential growth. Therefore we propose a reduction mechanism which prunes the state spaces on-the-fly, based on various different criteria.

The efficiency of the approach comes at the expense of expressiveness. Instead of representing the system's entire state space with all possible execution paths, we construct a *transient probability space* which allows for the computation of simple reachability properties only. We aim to address this limitation in our future work and investigate the verification of more complex temporal properties.

A more comprehensive description of the approach against the background of the three aforementioned scenarios will be given in the following sections. We start with some formal definitions.

5.2. *Formal definitions*

Definition 5.1 *A DTMC $\mathcal{M}$ is a tuple $(S, s_0, \mathcal{P}, AP, L)$ where S is a finite set of states, $s_0 \in S$ is the initial state, $\mathcal{P} \subseteq S \times S \rightarrow [0,1]$ is a transition probability function such that $\forall s \in S : \sum_{s' \in S} \mathcal{P}(s, s') = 1$ for each $s, s' \in S$, AP is a set of atomic propositions and $L : S \rightarrow 2^{AP}$ is a labelling function which assigns atomic propositions to states.*

Each state $s \in S$ represents one particular state of the agent, the probabilistic transitions $t \in (S \times S)$ can be viewed as the agent's *actions*, i.e. the choices an agent can make in order to change its state. The progression of the system over time spans a *transient probability space*.

Definition 5.2 *We denote with $\mathcal{T}_a(t)$ the transient probabilities of agent a over all states at time step t. With $\mathcal{T}_a(\phi, t)$ we denote the transient probability of agent a over a particular state ϕ at time step t.*

If the transition matrix T of DTMC $\mathcal{M}$ and the transient probabilities at time step 0 are known, then we can calculate the set of transient probabilities using the following iterative vector-matrix multiplication:

$$\mathcal{T}_a(i+1) = \mathcal{T}_a(i) \cdot T$$

Next we assume that we have multiple agents and that they perform their updates synchronously, i.e. they move into the next state *at the same time*. In this case, we can multiply the transient probabilities of the individual agents at time step t in order to determine the likelihood of all agents being in their respective states at time t. Using this method, we can now easily calculate the probability of an entire ABS being in a given state at a given point in time. Let ϕ be a desired global state of the ABS. We can subdivide ϕ into its agent-related sub-states and describe a sub-state with ϕ^i such that $\phi = \phi^0 \ \oplus \ \phi^1 \ \oplus \ ... \ \oplus \ \phi^{n-1}$, where $\oplus$ is some kind of concatenation operator[2]. If we want to determine the likelihood of a group of n agents being in state ϕ at time t, then we can simply multiply the individual transient probabilities of the agents at time t. This leads to the definition of global or system-wide transient probabilities.

Definition 5.3 *We denote with $\mathcal{T}_S(\phi, t) = \prod_{a=1}^{n} \mathcal{T}_a(\phi^a, t)$ the global or system-wide transient probabilities of a group of n agents at time t.*

Using the definition of global transient probabilities, we can now define the notion of reachability.

Definition 5.4 *A global state ϕ is reachable in system S if there is a time step t such that $\mathcal{T}_S(\phi, t) > 0$.*

For the formal specification of the reachability properties, we use a simple notation and borrow the $\Diamond$ (finally) operator from PCTL. We further include a notion of time by allowing to specify a temporal interval. Thus, $\Diamond^{[u,v]}\phi$ denotes that the system will reach state ϕ at some point between time steps u and v[3]. In cases where $u = v$ we simply write $\Diamond^{=u}\phi$. Since we are dealing with probabilistic systems, reachability properties need to be preceded by a probabilistic operator $\mathbb{P}$.

We can distinguish between *qualitative* and *quantitative* properties. A qualitative property describes a statement whose correctness is to be

2. We will illustrate this with an example in Section 5.3.
3. The specification of both state ϕ and temporal boundaries u and v are problem-dependant and need to be specified by the user during the verification process.

verified and its general form is $\mathbb{P}_{\bowtie p}(\Diamond^{[u,v]}\phi)$, where $\bowtie \in \{<,>,\leq,\geq\}$ and $p \in [0,1]$. Examples for qualitative properties are:

– $\mathbb{P}_{>0.5}(\Diamond^{=3}\phi)$: *"The probability of the ABS being in state ϕ after 3 time steps is higher than 50%.*

– $\mathbb{P}_{<0.1}(\Diamond^{[0,5]}\phi)$: *"Within the first five time steps, the probability of the ABS being in state ϕ is less than 10%."*

In a qualitative context, p describes the probability of reaching ϕ in the given time or interval. A qualitative property can thus be answered by calculating the transient probabilities as described above. This is formalised by the following satisfiability relation:

$$\begin{aligned} \mathbb{P}_{\bowtie p}(\Diamond^{[u,v]}\phi) = \text{true} \quad &\Leftrightarrow \quad \exists t : (u \leq t \leq v) \wedge (\mathcal{T}_S(\phi,t) \bowtie p) \\ \mathbb{P}_{\bowtie p}(\Diamond^{=u}\phi) = \text{true} \quad &\Leftrightarrow \quad \exists t : (t = u) \wedge (\mathcal{T}_S(\phi,t) \bowtie p) \end{aligned}$$

We can also specify *quantitative* properties whose general form is $\mathbb{P}_{=?}(\Diamond^{[u,v]}\phi)$. Instead of asking whether a statement holds with probability p and thus yielding a clear yes/no answer, quantitative properties ask for the value itself. The result is therefore either a single value (if $u = v$) or a set of values (if $u < v$). The connection between quantitative properties and transient probabilities can be described as follows:

$$\mathbb{P}_{=?}(\Diamond^{[u,v]}\phi) = p \Leftrightarrow p = \bigcup_{t=u}^{v} \mathcal{T}_S(\phi,t)$$

Examples for quantitative reachability probabilities are:

– $\mathbb{P}_{=?}(\Diamond^{=5}\phi)$: *"What is the probability of the ABS being in state ϕ after 5 time steps?"*

– $\mathbb{P}_{=?}(\Diamond^{[0,5]}\phi)$: *"What is the probability distribution of state ϕ for the first five time steps of the simulation?*

Our goal is to verify both types of transient reachability properties. More specifically, we want to (i.) prove that the ABS will or will not reach a given global system state, (ii.) prove that the ABS will reach a given global system state within probabilistic and temporal boundaries and (iii.) determine the probability that the ABS will reach a given global system state.

5.3. *Scenario 1: Transition matrix is known in advance*

As outlined in Section 5.1, we will analyse three different scenarios with increasing complexity. The first scenario assumes that the transition matrix of an agent is known in advance and agents are completely independent. We will relax these assumptions in subsequent sections.

In order to illustrate the approach, we introduce a simple ABS comprising 10 agents. For each agent, we use the simple finite state representation shown in Figure 1. We assume that all agents start in state A. In each state an agent makes a probabilistic choice about its next state. For simplicity reasons we assume that agents are completely independent, i.e. not interacting. As a consequence, the full DTMC of each agent and its transition matrix is known in advance and available to the verification algorithm. As described above, the verification approach consists of three steps. The first step involves the calculation of transient probabilities for all agents. Since we know the transition matrix and the initial probability distribution, we can easily obtain the transient probabilities of all agents using the vector-matrix multiplication described in Section 5.2. The results of this calculation are shown in Table 1 (left).

The second step involves the subdivision of the global desired state into agent-specific components. In the following we will represent a global state as a string of concatenated agent states. For example, $DDDAAAAAAA$ represents a system state where agents 1-3 are in local state D and all other agents are in local state A. At this point, we need a reachability question that we would like to answer. A global reachability question is often formulated in terms of a particular percentage of agents being in a given state, for example:

"What is the probability of the ABS reaching a state where at least 50% of the agents are in state D?"

Clearly there are numerous different global states which satisfy this requirement, e.g. $DDDDDDAAAA$, $ADDDDDDDDD$ etc. (ex-

	State			
	A	**B**	**C**	**D**
$\mathcal{T}_a(0)$	1	0	0	0
$\mathcal{T}_a(1)$	0.333	0.333	0.333	0
$\mathcal{T}_a(2)$	0.111	0.278	0.278	0.333
$\mathcal{T}_a(3)$	0.037	0.176	0.176	0.611
$\mathcal{T}_a(4)$	0.012	0.100	0.100	0.787
$\mathcal{T}_a(5)$	0.004	0.054	0.054	0.887

$\mathcal{T}_S(\phi, 0)$	$0^6 \cdot 1^4 = 0$
$\mathcal{T}_S(\phi, 1)$	$0^6 \cdot 0.333^4 = 0$
$\mathcal{T}_S(\phi, 2)$	$0.333^6 \cdot 0.111^4 \approx 2.07 \cdot 10^{-7}$
$\mathcal{T}_S(\phi, 3)$	$0.611^6 \cdot 0.037^4 \approx 9.75 \cdot 10^{-8}$
$\mathcal{T}_S(\phi, 4)$	$0.787^6 \cdot 0.012^4 \approx 4.93 \cdot 10^{-9}$
$\mathcal{T}_S(\phi, 5)$	$0.887^6 \cdot 0.004^4 \approx 1.25e \cdot 10^{-10}$

Table 1: Agent-specific (left) and system-wide (right) transient probabilities for the first five time steps (values are rounded)

actly 81,922 possible combinations)[4]. We denote with Φ the set of all possible combinations. We can now formally describe a final property to prove. We choose $\mathbb{P}_{>0.1}(\Diamond^{[0,5]}\Phi)$ which states that *the probability of the ABS being in any state $\phi \in \Phi$ within the first 5 time steps is higher than 10%.* In order to verify this property, we need to perform the verification separately for each $\phi \in \Phi$. In the following, we will describe the process for one randomly selected state $\phi = DDDDDDAAAA$ in which six agents are in state D and four agents are in state A. We first need to subdivide ϕ into its agent-specific components (with ϕ^i we denote the component of ϕ that represents the state of agent i): $\phi^0 = D$, $\phi^1 = D$, $\phi^2 = D$, $\phi^3 = D$, $\phi^4 = D$, $\phi^5 = D$, $\phi^6 = A$, $\phi^7 = A$, $\phi^8 = A$, $\phi^9 = A$. We can now perform the actual reachability analysis. First, we need to calculate the transient probabilities of the system for the five time steps as described in Section 5.2. The general formula is $n^6 \cdot m^4$ where n denotes the probability of state D and m denotes the probability of state A in the respective time step. Since all agents start in state A, the initial probability is $0^6 \cdot 1^4 = 0$, the probability at tick 1 is $0^6 \cdot 0.333^4$, and so on. The resulting values up to time step 5 are shown in Table 1 (right).

In order to determine the final result, we need to repeat this process for all 81,921 remaining states $\phi_i \in \Phi$ and calculate the sum of the resulting probabilities. We will denote with $\mathcal{T}_S(\Phi, t)$ the sum of the transient probability of all 81,922 states $\phi_i \in \Phi$ at time step t:

4. The number of possible combinations is subject to combinatorial explosion and therefore currently represents a serious bottleneck. We plan to address this problem in the future.

$\mathcal{T}_S(\Phi, 0)$	0.0
$\mathcal{T}_S(\Phi, 1)$	0.0
$\mathcal{T}_S(\Phi, 2)$	0.213
$\mathcal{T}_S(\Phi, 3)$	0.852
$\mathcal{T}_S(\Phi, 4)$	0.991
$\mathcal{T}_S(\Phi, 5)$	1.0

Table 2: Transient probabilities of being in any subset of Φ (values are rounded)

$$\mathcal{T}_S(\Phi, t) = \sum_{i=0}^{|\Phi|} (\mathcal{T}_s(\phi_i, t))$$

With the resulting list of transient probabilities (see Table 2), we can now easily prove the correctness of property $\mathbb{P}_{>0.1}(\Diamond^{[0,5]}\phi)$ by determining the maximum probability within the list:

$$max(\mathcal{T}_s(\Phi, t)) \approx 1.0$$

The results show that property $\mathbb{P}_{>0.1}(\Diamond^{[0,5]}\phi)$ almost surely holds since the probability of being in any of the states of Φ is $\approx 100\%$. Its correctness has thus been proven.

The example shows that the incremental nature of the approach can also help to answer verification questions very efficiently. For instance, in the example given above we could have stopped the verification in time step 2. At this point, the transient probability of being in any subset of Φ is 0.213 and the property thus holds. This also shows how the incremental nature also helps to answer inverse question like finding the *minimum time step* in which the property holds (2 in the example).

In the following subsections we relax the assumptions that we made before and outline ideas for (i.) the iterative construction of agent-specific state spaces and the calculation of transient probabilities in cases where the transition matrices are not known in advance and (ii.) the consideration of agent interaction.

Algorithm 1 Iterative model creation

```
1: for $0 \leq i < maxTime$ do
2:     for all $a \in Agents$ do
3:         for all $s \in States_A(a, i)$ do
4:             $succ \leftarrow reduce(getSucc(a, s))$
5:             for all $s' \in succ$ do
6:                 $\mathcal{T}(s', i+1) \leftarrow \mathcal{T}(s', i+1) + \mathcal{T}(s, i) \cdot \mathcal{P}(s, s')$
7:             end for
8:             $insert(States_A(a, i+1), succ)$
9:         end for
10:    end for
11: end for
12: return $\mathcal{T}$
```

5.4. *Scenario 2: Iterative calculation*

The example given in the previous section was based on the assumption that the DTMC and the transition matrix of a single agent is known in advance. This, however, will rarely be the case for a real-world ABS, due to the following reasons:

– The transition matrix of a single agent with unbounded variables can be huge (or even infinite)

– The transition probabilities at a particular time step are often influenced by the state of an agent's environment at that point in time (e.g. due to interaction between agents as discussed in Section 5.5)

As a solution to this problem we propose an iterative algorithm which, starting from an initial state s_0, expands the agent's state space and transition matrix continuously and calculates the transient probabilities on-the-fly until the maximum time step has been reached. The iterative model construction algorithm is outlined in Alg. 1. $States_A(a, t)$ denotes the possible states of agent a at time step t, $\mathcal{T}(s, t)$ to denotes the transient probabilities of state s at time step t and $\mathcal{P}(s, s')$ denotes the (single-step) transition probability between states s and s'.

The algorithm starts with the agent's initial state and uses a function $getSucc$ which accepts as input the current state and returns all possible successor states using the agent's behavioural rules. In order to alleviate combinatorial explosion, the number of successor states can be constrained through a reduction function (see line 5). The actual characteristic of the reduction function may vary and depend on the problem domain. A possible criterion could be the likelihood of a state, i.e. its transient probability. In each time step and for each successor state, the algorithm determines the transient probability by multiplying the transient probability of the current state with the transition probability between the current state and the successor state. Since a state can have multiple predecessor states, its transient probabilities are calculated in a cumulative way. Finally, we insert the temporary set of successor states into the overall set of states $States_A$ in order to be able to continue with the next time step. After the iterative process has finished, the actual verification can take place as described in Section 5.3.

An example which also includes the iterative calculation will be given in the next section. We will also outline our idea of how dependencies between agents due to interaction can be integrated into the iterative model construction process described in this section.

5.5. *Scenario 3: Introducing agent interaction*

So far we assumed that agents are separate entities which are completely independent from each other. However, one of the characteristic features of agents is their ability to communicate, to coordinate actions and to exchange information. Consider e.g. a disease transmission simulation where each agent's probability of changing its current state (e.g. from "healthy" to "infected") is influenced by the states of its neighbours. The ability to interact introduces significant dependencies which prevent us from treating agents completely separately during the model construction process. However, by slightly modifying the iterative process described in the previous section, we can take into account an agent's environment which makes it possible to react to changes and to adapt the transition probabilities accordingly. Instead of determining the successor states together with the transient probabilities independently for each agent and for a number of time steps in advance, we introduce a *perception step* in the beginning of each tick (line 3 in Algorithm 2). During this step, each agent perceives its current environment and adapts its behaviour, i.e. its transition probabilities, accordingly.

Algorithm 2 Introducing perception

```
1: for $0 \leq i < maxTime$ do
2:     for all $a \in Agents$ do
3:         $perceive()$
4:         for all $s \in States_A(a, i)$ do
5:             $succ \leftarrow reduce(getSucc(a, s))$
6:             for all $s' \in succ$ do
7:                 $\mathcal{T}(s', i+1) \leftarrow \mathcal{T}(s', i+1) + \mathcal{T}(s, i) \cdot \mathcal{P}(s, s')$
8:             end for
9:             $insert(States_A(a, i+1), succ)$
10:        end for
11:    end for
12: end for
13: return $\mathcal{T}$
```

In order to illustrate this principle, we use the simple disease transmission simulation mentioned above as our example scenario. In this model, each agent can be either in state I (infected) or H (healthy) at any point in time. As before, state transitions are probabilistic (see Figure 3). A distinctive feature of agents is *autonomy* and therefore we assume that, rather than acting purely reactively to its environment, an agent makes its decisions primarily based on its own internal state. However, due to the agent's ability to perceive its environment and act accordingly, the state of the neighbourhood still influences its actual transition probabilities. Therefore we assume that the transition rates in our example can be subdivided into an *internal* and an *external* factor:

$$probRec \quad = \quad int_1 + ext_1 \cdot \left(\frac{numHealthyNB}{numNB} \right) \tag{1}$$

$$probInf \quad = \quad int_2 + ext_2 \cdot \left(\frac{numInfNB}{numNB} \right) \tag{2}$$

$numHealthyNB$ denotes the number of healthy neighbours, $numInfNB$ denotes the number of infected neighbours and $numNB$ denotes the total number of neighbours. Equation 1 denotes the *recovery rate*, i.e. the transition probability from I to H, and equation 2 denotes the *infection rate*, i.e. the transition probability from H to I. By varying int_i and ext_i, we can adjust the ratio between an agent's autonomy and its reactivity, i.e. the extent to which environmental factors

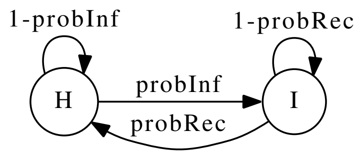


Figure 3: DTMC representation of an agent in the disease transmission simulation

influence its decision making process. For example, if we set $int_1 = 0.7$ and $ext_1 = 0$, then the agent is completely autonomous, i.e. it has a 70% chance of recovering after being infected, regardless how many infected agents there are in its neighbourhood. In contrast, if we set $int_1 = 0$ and $ext_1 = 1$, then the agent is purely reactive and its probability of recovering solely depends on how many healthy agents there are in its neighbourhood. The level of autonomy that an agent exhibits has a significant influence on the accuracy of our compositional verification process. We will describe this issue in the following paragraphs.

It is obvious that in this scenario an agent's DTMC can not be designed in advance since the actual transition rates at time t depend on the status of its environment at t. However, using the iterative construction method described in Section 5.4, we can take into account the status of the environment and still construct the agent's transient probabilities separately without having to construct an exhaustive model of the system as follows: At each time step and within the function $perceive$, we can easily determine the probability of the neighbours of agent a being infected at the same time by looking at their transient probabilities. Using this information, we can then adapt the transition probabilities of agent a to reflect the status of the neighbourhood.

For simplicity reasons, we limit the size of our ABS to 3 agents and refer to them in the equations with $a1$, $a2$ and $a3$. We further assume that agent 1 is initially infected and agents 2 and 3 are initially healthy. The initial transient probabilities of the agents are thus as follows:

$$\begin{aligned}\mathcal{T}_{a1}(0) &= \{H=0, I=1\}\\ \mathcal{T}_{a2}(0) &= \{H=1, I=0\}\\ \mathcal{T}_{a3}(0) &= \{H=1, I=0\}\end{aligned}$$

We set $int_1 = 0.6$, $ext_1 = 0.2$, $int_2 = 0.2$ and $ext_2 = 0.2$. Based on this information, we can now calculate the transition probabilities of each agent. For the first time step, the calculation is straightforward since the initial probabilities of each agent being in a particular state are absolut, i.e. 100%:

$$\begin{aligned}tI\_H_{a1} &= 0.4\\ tH\_I_{a1} &= 0.6\end{aligned}$$

$$\begin{aligned}tI\_H_{a2} &= 0.3\\ tH\_I_{a2} &= 0.7\end{aligned}$$

$$\begin{aligned}tI\_H_{a3} &= 0.3\\ tH\_I_{a3} &= 0.7\end{aligned}$$

The transient probabilities of all agents for time step 1 are thus as follows:

$$\begin{aligned}\mathcal{T}_{a1}(1) &= \{H=0.4, I=0.6\}\\ \mathcal{T}_{a2}(1) &= \{H=0.3, I=0.7\}\\ \mathcal{T}_{a3}(1) &= \{H=0.3, I=0.7\}\end{aligned}$$

The calculation of transition probabilities for the next time step is slightly more complicated since agents 2 and 3 now both have a chance of being either healthy or infected. We start with the calculation of the transition probabilities for agent 1. Both of its neighbours have a 30% chance of being in state H and a 70% chance of being in state I. Let A_iB_j denote a situation where agent i is in state A and agent j is in state B. Let further denote $Pr(A_iB_j)$ the probability of A_iB_j. In time step 1, the following combinations are now possible for the neighbourhood of agent 1:

$$\begin{aligned} Pr(H_2H_3) &= 0.09 \quad (3)\\ Pr(H_2I_3) &= 0.21 \quad (4)\\ Pr(I_2H_3) &= 0.21 \quad (5)\\ Pr(I_2I_3) &= 0.49 \quad (6) \end{aligned}$$

Since we are not interested in *which* agent but rather in *how many* agents are in a particular state, we can compress this list using *counter abstraction* by replacing the exhaustive list of possible global states with their *generic representatives* [DMP09, ET99]. Let $\Phi = (H_1, H_2, I_3)$ denote a global state where agents 1 and 2 are healthy and agent 3 is infected. Let further $\Psi = (H_1, I_2, H_3)$ denote a global state where agents 1 and 3 are healthy and agent 2 is infected. From the perspective of counter abstraction, both situations are symmetrically equivalent since the number of agents being healthy and the number of agents being infected are the same. Φ and Ψ thus refer to the same abstract state and can be replaced by the generic representative $(2H, 1H)$ which states that "two agents are healthy, one is infected". Using this technique, we can now aggregate equations 3 to 6 as follows:

$$\begin{aligned} Pr(1H, 1I) &= 0.42\\ Pr(0H, 2I) &= 0.09\\ Pr(2H, 0I) &= 0.49 \end{aligned}$$

Using these global representatives, the transition probabilities can now be calculated as follows:

	Agent 1	
Time	**H**	**I**
$\mathcal{T}_a(0)$	0	1
$\mathcal{T}_a(1)$	0.4	0.6
$\mathcal{T}_a(2)$	0.26	0.74
$\mathcal{T}_a(3)$	0.254	0.746
$\mathcal{T}_a(4)$	0.251	0.749
$\mathcal{T}_a(5)$	0.250	0.750

	Agent 2	
Time	**H**	**I**
$\mathcal{T}_a(0)$	1	0
$\mathcal{T}_a(1)$	0.3	0.7
$\mathcal{T}_a(2)$	0.27	0.73
$\mathcal{T}_a(3)$	0.253	0.747
$\mathcal{T}_a(4)$	0.251	0.749
$\mathcal{T}_a(5)$	0.250	0.750

	Agent 3	
Time	**H**	**I**
$\mathcal{T}_a(0)$	1	0
$\mathcal{T}_a(1)$	0.3	0.7
$\mathcal{T}_a(2)$	0.27	0.73
$\mathcal{T}_a(3)$	0.253	0.747
$\mathcal{T}_a(4)$	0.251	0.749
$\mathcal{T}_a(5)$	0.250	0.750

Table 3: Transient probabilities for time steps 0 to 5 (values are rounded)

$$\begin{aligned} tI\_H &= 0.6 \cdot Pr(0H, 2I) \\ &\quad +(0.6 + 0.1) \cdot Pr(1H, 1I) \\ &\quad +(0.6 + 0.2) \cdot Pr(2H, 0I) = 0.74 \\ tH\_I &= (0.2 + 0.2) \cdot Pr(0H, 2I) \\ &\quad +(0.2 + 0.1) \cdot Pr(1H, 1I) \\ &\quad +0.2 \cdot Pr(2H, 0I) = 0.26 \end{aligned}$$

For agents 2 and 3, the transition probabilities can be calculated accordingly and used to determine the transient probabilities for $t = 2$:

$$\begin{aligned} \mathcal{T}_{a1}(2) &= \{H = 0.26, I = 0.74\} \\ \mathcal{T}_{a2}(2) &= \{H = 0.27, I = 0.73\} \\ \mathcal{T}_{a3}(2) &= \{H = 0.27, I = 0.73\} \end{aligned}$$

This process can be repeated until the final time step has been reached (for a full list of transient probabilities until time step 5, see Table 3). In this way, we can calculate the local transient probabilities for each agent in an iterative and incremental way and take into account any dependencies between agents which might influence their transition probabilities. After the transient probabilities for each agent have been calculated, the actual verification can then be performed as described for the simple scenario in Section 5.3.

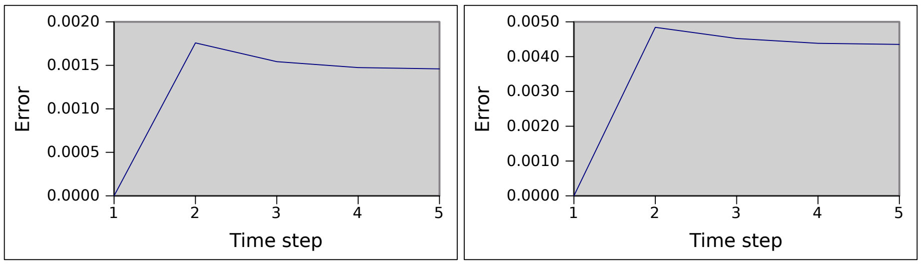


Figure 4: Errors between compositional and non-compositional calculation of transient probabilities for global state HHH (left) and III (right)

As mentioned before, the level of autonomy that an agent exhibits has a significant influence on the accuracy of the compositional verification process. If an agent's decision at time point t depends on its neighbours' states at t, this dependency will continue to hold for all future states. These *path dependencies* between the agents' life histories influence the amount to which the likelihood of a global system state can be determined by multiplying the transient probabilities of individual agents. The more an agent is dependent upon the state of its environment (i.e. the more reactive it is), the more path dependencies are introduced over time and, as a consequence, the less accurate the results of the described approach will be. In our disease transmission example, the ratio between an agent's autonomy and its reactive nature is determined by the values of int_i and ext_i in the calculation of transition probabilities. The inaccuracy, i.e. the numerical difference between the results of the compositional calculation for the global states HHH and III and the original values derived from an exhaustive model, is shown in Figure 4. Since autonomy is a distinctive feature of the concept of agency, we assume that the level of autonomy will typically be high in ABSs. Nevertheless, the ratio between the error introduced through increased reactivity and the accuracy of the verification results is an important factor for the applicability of our approach which we aim to investigate in further detail in the future.

5.6. *Evaluation*

At the current stage, we have limited our evaluation to the simple non-interacting scenario described in Section 5.3. In order to compare the compositional approach described in this paper with reachability analysis using conventional model checking, we used PRISM [KNP02], a popular probabilistic model checker, to prove $\mathbb{P}_{>0.1}(\Diamond^{\leq 5}\phi)$. Before performing the actual verification, PRISM needs to construct the state space. In the case of the example above, the resulting space of the ABS (10 agents, 4 states per agent, synchronous update) comprises $\approx 10^{6.02}$ states and $\approx 10^{9.03}$ transitions. Determining the probability for reaching a single $\phi \in \Phi$ (we chose our global example state $DDDDDDAAAA$) took around 0.4 seconds on a custom laptop with Intel® Core™ i5 processor (with four 2.27 GHz cores), 4GB of memory and Ubuntu 10.10 as operating system. In order to prove the property, it is necessary to determine the probabilities for all $\phi \in \Phi$. If we follow a naïve approach and simply execute PRISM repeatedly for all 81,922 elements in Φ, this amounts to around 9 hours in total. In contrast, the execution of our verification approach for all 81,922 states took around 19 seconds in total. The difference in runtime is significant, but we need to keep in mind that, as opposed to PRISM, the approach described in this paper is limited to the verification of transient properties and does not allow for the verification of more complex reachability probabilities, let alone to provide support for full PCTL. However, if the verification questions can be formulated in terms of transient probabilities, then the described approach can provide an efficient and robust alternative to conventional model checking.

6. Conclusion and future work

In this paper we investigated model checking-based verification against the background of large-scale ABSs. We identified a number of ABS-specific properties together with recommendations for their verification. Following these recommendations, we presented the first sketch of a possible verification technique based on reachability analysis over transient probabilities. The approach is iterative and compositional and allows to calculate the probabilities incrementally. First experiments showed that, using this approach, the correctness of transient reachability properties can be verified quickly, even for large-scale ABSs. The nature of the reachability property plays an important role for the efficiency of the algorithm. As described in Section 5.3, we still face combinatorial explosion when dealing with properties that refer to percentages of agents being in a given state. In this case, we have to deal with a number of possible state combinations that quickly becomes unmanageable. We plan to address this issue e.g. by applying appropriate approximation techniques. Due to the focus on transient probabilities, we are currently limited to a small subset of reachability properties. We plan to extend the approach towards the support of more expressive probabilistic temporal logics like PCTL for the formulation of temporal properties. With respect to agent interaction, we outlined our idea of taking into account an agent's neighbourhood during the iterative creation of the transient probabilities by means of an additional perception step. However, if interaction is frequent and the level of reactivity is high, compositional verification results get increasingly inaccurate. We plan to investigate this issue further and develop a stronger understanding about the ratio of reactivity and accuracy. Furthermore, we aim to elaborate on the concept of compositional verification by taking into account topological characteristics (e.g. the community structure) of the underlying agent network and performing verification in a bottom-up way for large-scale ABSs with hundreds and thousands of agents.

Acknowledgements

We would like to thank the anonymous reviewers, the audience of the V2CS'11 workshop in Paris as well as the members of the verification research group at Oxford University for their helpful comments and suggestions. We would also like to thank Sandtable Ltd. in London for supporting this research.

References

[Axe86] R. Axelrod. An evolutionary approach to norms. *American Political Science Review*, 80(4):1095–1111, 1986.

[BdA95] A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In P. Thiagarajan, editor, *Proc. 15th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, volume 1026 of *LNCS*, pages 499–513, Berlin, Heidelberg, 1995. Springer.

[BFVW06] R. H. Bordini, M. Fisher, W. Visser, and M. Wooldridge. Verifying multi-agent programs by model checking. *Autonomous Agents and Multi-Agent Systems*, 12:239–256, March 2006.

[BFW09] P. Ballarini, M. Fisher, and M. Wooldridge. Uncertain agent verification through probabilistic model-checking. In *Safety and Security in Multiagent Systems*, volume 4324 of *Lecture Notes in Computer Science*, pages 162–174. Springer, Berlin, Heidelberg, 2009.

[BML$^+$06] R. A. Belecheanu, S. Munroe, M. Luck, T. Payne, T. Miller, P. McBurney, and M. Pěchouček. Commercial applications of agents: lessons, experiences and challenges. In *Proc. 5th Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'06)*, AAMAS '06, pages 1549–1555, New York, NY, USA, 2006. ACM.

[Bon02] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. In *Proc. National Academy of Sciences of the United States of America*, volume 99, pages 7280–7287, May 2002.

[CBRZ01] E. Clarke, A. Biere, R. Raimi, and Y. Zhu. Bounded model checking using satisfiability solving. *Formal methods in system design*, 19:7–34, July 2001.

[CEJS98] E. Clarke, E. Emerson, S. Jha, and A. Sistla. Symmetry reductions in model checking. In Alan Hu and Moshe Vardi, editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 147–158. Springer, Berlin, Heidelberg, 1998.

[CGJ$^+$01] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Progress on the state explosion problem in model checking. In Reinhard Wilhelm, editor, *Informatics*, volume 2000 of *Lecture Notes in Computer Science*, pages 176–194. Springer, Berlin, Heidelberg, 2001.

[CGMP99] E. Clarke, O. Grumberg, M. Minea, and D. Peled. State space reduction using partial order techniques. *International Journal on Software Tools for Technology Transfer (STTT)*, 2:279–287, 1999.

[Cla97] E. Clarke. Model checking. In S. Ramesh and G. Sivakumar, editors, *Foundations of Software Technology and Theoretical Computer Science*, volume 1346 of *Lecture Notes in Computer Science*, pages 54–56. Springer, Berlin, Heidelberg, 1997.

[Cou85] P. J. Courtois. On time and space decomposition of complex structures. *Communications of the ACM*, 28:590–603, June 1985.

[DDV08] M. I. Dekhtyar, A. J. Dikovsky, and M. K. Valiev. Temporal verification of probabilistic multi-agent systems. In A. Avron, N. Dershowitz, and A. Rabinovich, editors, *Pillars of Computer Science*, pages 256–265. Springer, Berlin, Heidelberg, 2008.

[DMP09] A. F. Donaldson, A. Miller, and D. Parker. Language-level symmetry reduction for probabilistic model checking. In *Proc. 6th. Int. Conf. on the Quantitative Evaluation of Systems (QUEST'09)*, pages 289 –298, sept. 2009.

[EA96] J. M. Epstein and R. L. Axtell. *Growing Artificial Societies: Social Science from the Bottom Up*, volume 1 of *MIT Press Books*. The MIT Press, June 1996.

[ET99] E. A. Emerson and R. J. Trefler. From asymmetry to full symmetry: New techniques for symmetry reduction in model checking. In *Proc. 10th IFIP WG 10.5 Advanced Research Working Conference on Correct Hardware Design and Verification Methods (CHARME'99)*, pages 142–156, London, UK, 1999. Springer-Verlag.

[FLW06] H. Fecher, M. Leucker, and V. Wolf. Don't know in probabilistic systems. In *Model checking software*, pages 71–88. Springer, 2006.

[HJ94] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[HLMP04] T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *Proc. 5th Int. Conf. on Verification, Model Checking and Abstract Interpretation (VMCAI'04)*, volume 2937 of *LNCS*, pages 307–329, Berlin, Heidelberg, 2004. Springer.

[HQR98] T. A. Henzinger, S. Qadeer, and S. K. Rajamani. You assume we guarantee: Methodology and case studies. In *Proc. 10th Int. Conf. on Computer Aided Verification (CAV'08)*, pages 440–451, London, UK, 1998. Springer.

[IIGS09] L. R. Izquierdo, S. S. Izquierdo, J. M. Galán, and J. I. Santos. Techniques to understand computer simulations: Markov chain analysis. *Journal of Artificial Societies and Social Simulation*, 12(1):6, 2009.

[KDF10] S. Konur, C. Dixon, and M. Fisher. Formal verification of probabilistic swarm behaviours. In M. Dorigo, M. Birattari, G. Di Caro, R. Doursat, A. Engelbrecht, D. Floreano, L. Gambardella, R. GroSS, E. Sahin, H. Sayama, and T. Stützle, editors, *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 440–447. Springer, Berlin, Heidelberg, 2010.

[KNP02] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In Tony Field, Peter Harrison, Jeremy Bradley, and Uli Harder, editors, *Computer Performance Evaluation: Modelling Techniques and Tools*, volume 2324 of *Lecture Notes in Computer Science*, pages 113–140. Springer, Berlin, Heidelberg, 2002.

[LMP04] M. Luck, P. McBurney, and C. Preist. A manifesto for agent technology: Towards next generation computing. *Autonomous Agents and Multi-Agent Systems*, 9:203–252, November 2004.

[LQR09] A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In Ahmed Bouajjani and Oded Maler, editors, *Computer Aided Verification*, volume 5643 of *Lecture Notes in Computer Science*, pages 682–688. Springer, Berlin, Heidelberg, 2009.

[Mar07] R. E. Marks. Validating simulation models: A general framework and four applied examples. *Computational Economics*, 30:265–290, October 2007.

[McM92] K. L. McMillan. *Symbolic model checking: an approach to the state explosion problem*. PhD thesis, Pittsburgh, PA, USA, 1992. UMI Order No. GAX92-24209.

[MMK07] D. Midgley, R. E. Marks, and D. Kunchamwar. Building and assurance of agent-based models: An example and challenge to the field. *Journal of Business Research*, 60(8):884 – 893, 2007. Complexities in Markets Special Issue.

[New06] M. E. Newman. Modularity and community structure in networks. *Proc. National Academy of Sciences of the United States of America*, 103(23):8577–8582, June 2006.

[PSR98] H. V. D. Parunak, R. Savit, and R. L. Riolo. Agent-based modeling vs. equation-based modeling: A case study and users' guide. In *Proc. 1st Int. Workshop on Multiagent Systems and Agent-Based Simulation (MABS'98)*, pages 10–25, London, UK, 1998. Springer.

[Sar08] R. G. Sargent. Verification and validation of simulation models. In *Proc. 40th Conf. on Winter Simulation*, WSC '08, pages 157–169. Winter Simulation Conference, 2008.

[Sch69] T. C. Schelling. Models of segregation. *American Economic Review*, 59(2):488–93, May 1969.

[WBBH11] W. Wan, J. Bentahar, and A. Ben Hamza. Model checking epistemic and probabilistic properties of multi-agent systems. In *Modern Approaches in Applied Intelligence*, volume 6704 of *Lecture Notes in Computer Science*, pages 68–78. Springer, Berlin, Heidelberg, 2011.