Towards quantitative analysis of multiagent systems through statistical model checking

Benjamin Herd, Simon Miles, Peter McBurney, and Michael Luck

Department of Informatics, King's College London, United Kingdom

Abstract. Due to their immense complexity, large-scale multiagent systems are often unamenable to exhaustive formal verification. Statistical approaches that focus on the verification of individual traces can provide an interesting alternative. However, due to its focus on finite execution paths, trace-based verification is inherently limited to certain types of correctness properties. We show how, by combining sampling with the idea of trace fragmentation, statistical model checking can be used to answer interesting quantitative correctness properties about multiagent systems on different observational levels. We illustrate the idea with a simple case study from the area of swarm robotics.

Keywords: verification, statistical model checking, multiagent systems, quantitative analysis

1 Introduction

Due to their distributed nature and their capability to exhibit emergent behaviour, multiagent systems can be hard to engineer and to understand. Similar to other software systems, however, questions of correctness arise and verification plays an important role. Formal verification aims to answer correctness questions in a rigorous and unambiguous way. Temporal logic model checking, for example, aims to find an accurate solution to a given correctness property property by exhaustively searching the state space underlying the system under consideration (the *model*) and thus exploring all possible execution paths [1]. This approach is, in general, only feasible if the state space of the model is of manageable size. In the presence of non-determinism which may, for example, arise from the different possible interleavings of individual agent actions or from uncertainty w.r.t. the representation of individual agent behaviours, the state space may grow exponentially which renders formal exhaustive verification infeasible for non-trivial systems. This exponential blow-up in the number of states is a well-known problem and commonly referred to as 'state space explosion'.

One interesting way to circumvent combinatorial explosion that works for probabilistic systems is to use a *sampling approach* and employ statistical techniques to obtain approximate verification results. In this case, n finite execution paths or *traces* are sampled from the underlying state space and a property ϕ is checked on each of them. By increasing the number of traces that ϕ is checked on, the probability of ϕ can be estimated to the desired level of precision. Techniques for statistical inference, e.g. *hypothesis testing*, can then be used to determine the significance of the results. Approaches of this kind are summarised under the umbrella of *statistical model checking*

[11]. Due to its approximate nature, statistical model checking allows for the verification of large-scale (or even infinite) systems in a timely manner. Traces are typically obtained through *simulation*, either by repeatedly executing an existing real-world system, or by 'unrolling' a formal state transition representation of a system for a certain number of time steps (as in the case of statistical model checking).

Consider, for example, a robot swarm whose efficiency is defined by its emergent collective behaviour. Due to the high level of interconnectivity and the global focus, it is not sufficient to verify individual robots in isolation. On the other hand, aspects such as a high level of heterogeneity, a complex environment, or simply an interest in the individual behaviours may also render the application of pure macro-level verification insufficient or even impossible. In this case, statistical verification represents an interesting alternative. However, because of its focus on finite execution paths, trace-based verification is inherently limited to linear time properties and lacks some of the quantitative capabilities of its non-statistical counterpart [9]. For example, due to the lack of branching information, properties about the transition behaviour are not verifiable in a trace-based context. Furthermore, statistical verification generally ignores the internal structure of the traces which limits its use for complex multiagent systems.

On the other hand, the statistical approach provides interesting opportunities. In this paper, we present our research efforts with respect to the aforementioned problems by showing how trace-based verification in combination with statistical analysis can be used to answer interesting quantitative correctness properties about multiagent systems. The contributions of this paper can be summarised as follows.

1. In Section 3, we show that simulation traces of multiagent systems represent *sets of sets of samples* obtained from *different sample spaces*, the choice of which depends on the question to be answered. We formally introduce the notion of *trace fragments* and describe how they correspond with fine-grained correctness properties. We also introduce the idea of *in-trace sampling*.
2. In Section 4, we introduce a simple specification language in order to illustrate the requirements that a specification language needs to satisfy in order to allow for the formulation of properties about multiagent systems on different observational levels. Making use of in-trace sampling, the language also allows for the formulation of properties about the *average behaviour* of individual agents.
3. In Section 5, we show how a combination of trace fragmentation and statistical verification can be used to estimate *residence probabilities* and *transition probabilities*, and to detect *correlations* between different types of events.

The usefulness of quantitative analysis is illustrated with a small case study from the area of swarm robotics in Section 6.

2 Related work

Whilst the classical, non-probabilistic approach to model checking produces a clear yes/no answer to a given correctness property, quantitative analysis aims to use verification techniques to produce numeric insights into the system under consideration, e.g. *transition probabilities*, *costs* or *rewards*. It is thus not surprising that quantitative analysis forms an important part of probabilistic approaches to model checking.

PRISM [10], for example, the most widely used probabilistic model checker, allows for the verification of a wide range of quantitative properties, among them best, worst, and average-case system characteristics [9]. PRISM uses BDD-based symbolic model checking and allows for the verification of properties formulated in a variety of different logics — among them probabilistic versions of Computation Tree Logic (CTL) and Linear Temporal Logic (LTL), as well as Continuous Stochastic Logic (CSL) — on different types of models, e.g. Discrete-Time (DTMC) and Continuous-Time Markov Chains (CTMC) and Markov Decision Processes (MDP). It is, for example, possible to integrate costs and rewards into the verification process which allows for the formulation of properties about *expected* quantities, e.g. the "expected time", or the "expected number of lost messages". Due to its exhaustive nature, PRISM generally suffers from the same combinatorial issues as other non-probabilistic model checkers. In order to circumvent this problem, it also allows for simulation-based (i.e. trace-based) verification using different statistical model checking approaches [13]. In this context, both conventional probabilistic linear time properties, i.e. $P_{=?}(\phi)$, and reward-based properties, i.e. $R_{=?}(\phi)$, can be answered. At the current stage, PRISM views traces as monolithic entities and does not exploit their internal structure. This limits its usefulness for the verification of complex multiagent systems.

Apart from the work on simulation-based verification using PRISM, quantitative analysis in the context of trace-based verification has been largely neglected to date. An interesting idea has been presented by Sammapun *et al.* [14]. The authors propose a trace decomposition based on the idea of *repetitive behaviour*. The decomposition is performed by means of conditional probabilities. This is then extended with hypothesis testing in order to determine the confidence in the estimation. As opposed to our approach which assumes the presence of a (possibly large) number of individual sample traces, the work of Sammapun *et al.* is focussed on a pure runtime verification setting in which only one consistently growing trace is available. The decomposition is used to obtain from the runtime trace a number of individual sample traces which are then used to answer conventional probabilistic linear-time properties such as done by PRISM.

A related approach has been presented by Finkbeiner *et al.* [3]. They propose an extension of LTL which allows for the formulation of additional statistics over traces, e.g. the *"average number of X"* or *"how often does X happen"*. Similar to the work of Sammapun *et al.*, they focus on a single trace obtained by observing a running system.

3 Events, properties, and their probability

In the previous sections, we used the term 'events' loosely when speaking about the formulation of properties. The purpose of this section is to give a formal definition for the notion of events and their association with formulable correctness properties.

3.1 Structure and probability of simulation traces

The purpose of this section is to formally associate the set of traces of a multiagent system obtained through simulation with a *probability space*. This allows us to talk about *events* and their *probability*. We show that, by varying the set of outcomes that one focusses on, events of different granularity become detectable.

Let us start with a formal representation of our multiagent system. We are not concerned with advanced modalities like knowledge or strategies here, so we assume that the state of an individual agent is defined as a simple set of attributes and their values. The state space of the multiagent system can then be described as a simple state transition system. Let S_i = denote the set of states of agent i. For n agents, $S \subseteq S_1 \times S_2 \times ...S_n$ then denotes the set of global states[1]. We assume that the multiagent system is probabilistic in nature, i.e. in the presence of multiple successor states, a probabilistic choice about which state the system transitions into will be made. Let therefore $P : S \times S \rightarrow [0, 1]$ be a *probabilistic transition function* such that $\forall\, s : S \bullet \sum_{s' \in S} P(s, s') = 1$. The multiagent system can then be described formally as a *probabilistic transition system* $\mathcal{M} = (S, P, s_0)$ where $s_0 \in S$ is the initial state. We denote each possible finite path $\omega = \langle s_0, s_1, ..., s_k \rangle$ of length k through $\mathcal{M}$ as a *simulation trace*. In the presence of individual agents, simulation traces have an internal structure. Given n agents, a simulation trace can be subdivided into n *agent traces*.

In the presence of transition probabilities, it is intuitively clear that each simulation trace ω occurs with a certain probability, denoted $Pr(\omega)$, which is the product of all individual transition probabilities:

$$Pr(\omega) = P(s_0, s_1) \cdot P(s_1, s_2) \cdot ... \cdot P(s_{n-1}, s_n) = \prod_{0 \leq i < n} P(s_i, s_{i+1}) \qquad (1)$$

In the presence of long simulation runs, restricting the focus of attention to the probability of full traces may be too coarse-grained. Traces represent (possibly long) sequences of system states which themselves also have a complex internal structure; in the course of a simulation run, numerous *events* take place which constitute themselves as changes to the state of the system. A trace represents all the states of the underlying run and can thus be seen as a rich source of analysis. In addition to the probability of the trace itself, it is therefore useful to also determine the probability of all individual events represented by it. However, in order to talk about events and their probability, we first need to make sets of traces *measurable*. To this end, we associate a *probability space* with the set of simulation traces. A probability space is a triple (Ω, Σ, Pr) where Ω is the *sample space*, $\Sigma \subseteq \mathbb{P}\,\Omega$ is a σ-algebra and $Pr : \Sigma \rightarrow [0, 1]$ is a probability measure. The sample space Ω can be seen as the set representing all possible *outcomes* of an experiment. Imagine, for example, throwing a die. In this case, the sample space is $\Omega = \{1, 2, 3, 4, 5, 6\}$. We can now start to define possible events within the set of outcomes. A single event represents a set of outcomes which all satisfy a common criterion. For example, getting an even number when throwing a die is represented by the set $\{2, 4, 6\}$. Formally, the set of events forms a *σ-algebra* $\Sigma \subseteq \mathbb{P}\,\Omega$ on Ω, where Σ is a subset of the power set of Ω. A σ-algebra Σ also needs to satisfy the following requirements: (i) Σ contains the empty set $\varnothing$, (ii) Σ is closed under complements: if A is in Σ then so is its complement $\overline{A} = (\Omega \backslash A)$, and (iii) Σ is closed under countable unions: if $A_1, A_2, ...$ are in Σ then so is their union $A = \bigcup A_n$. Furthermore, in order to assign a probability with an event, we need a *probability measure* $Pr : \Sigma \rightarrow [0, 1]$ which is a function that assigns to each event $E \in \Sigma$ a number between 0 and 1. Pr also needs to satisfy the following requirements: (i) Pr is countably additive: for all

[1] For simplicity, we omit the environment in our formal description.

countable collections $A = \{A_1, A_2, ..., A_n\} \in \Sigma$, $Pr(\bigcup A_i) = \sum(Pr(A_i))$, and (ii) $Pr(\varnothing) = 0$ and $Pr(\Omega) = 1$. A probability space is then defined as a triple (Ω, Σ, Pr) comprising the sample space Ω, σ-algebra Σ and probability measure Pr. In the context of probability theory, the events $\omega \in \Sigma$ are said to be *measurable* [1].

3.2 Simulation and sampling: trace fragmentation

In order to talk about events in the context of simulation runs, the set Tr_s of simulation traces which a simulation can produce needs to be made measurable by associating it with a probability space. We start with the sample space. A trace obtained through a single simulation run (if properly randomised, which we assume here) can be seen as a single sample drawn from the *set of finite traces* as defined by the logic within the model. However, at the same time, a single trace of length k also represents a set of k samples drawn from the *set of states* defined by the model. Furthermore, it also represents a set of $\binom{k}{2}$ samples drawn from the *set of state tuples*, a set of $\binom{k}{3}$ samples drawn from the *set of state triples*, and so on. Even more, given n agents, each simulation trace also represents n samples drawn from the *set of agent traces*, each of which itself represents a set of k samples from the *set of agent states*, etc.

In general, the description of a probabilistic state-based model yields a large range of different sets of outcomes that one can draw from: a set of agent or group states (one for each possible group of agents), of agent or group state pairs, of agent or group state triples, etc. Each individual simulation run represents one or many samples from each of those sets. As described above, each set of outcomes corresponds with a different probability space and thus allows for the detection of different events. Just by interpreting the same outcome in different ways, different types of events become detectable.

It becomes clear that a single simulation trace already represents a rich source of analysis. Let us now briefly look at the types of outcomes that one is *typically* interested in. We can assume that, in a simulation context, we are mostly interested in events defined over *coherent trace fragments*, rather than over arbitrary tuples of states. Informally, a coherent trace fragment is any sequence of states which exists in the underlying state space. Fragments of length 1 represent individual states, fragments of length 2 represent states and their direct successors, fragments of length 3 represent states and their two subsequent states, etc. Formally, the set $\mathcal{F}_k$ of coherent trace fragments of length k is defined as the set of sub-sequences of states, i.e. sub-traces, of length k:

$$\mathcal{F}_k == \bigcup\nolimits_{\omega \in Tr_s} \{p \text{ in } \omega \mid \#p = k\} \tag{2}$$

Each fragment size represents a certain level of *granularity* with respect to the simulation outcome. Before defining the sample space of a simulation, it is therefore important to clarify the granularity necessary to answer a given question. For example, some questions are formulated over entire simulation traces, i.e. members of the set $\mathcal{F}_t$. Typical representatives of this group are temporal questions that involve statements like, for example, *eventually* or *always*. In this case, the set from which samples need to be drawn is the set of all full traces, i.e. $\Omega = Tr_s$. The σ-algebra Σ (the set of possible events defined as a subset of $\mathbb{P}\,\Omega$) thus represents the set of all possible sets of traces.

For other questions, a finer level of granularity is needed. Consider, for example, a question about the existence of a particular state transition. On a full simulation trace, the state transition of interest may occur several times. In order to detect all occurrences (and thus measure the event's probability), it is not sufficient to look at complete traces. Instead, we need to look at *trace fragments* of length 2, i.e. at tuples of immediately succeeding states drawn from the set $\mathcal{F}_2$. This is necessary since any state transition is described by its start and end state. If questions about the probability of a single agent attribute valuation are to be answered, i.e. questions about a particular property of an individual state, then the set that samples need to be drawn from is the set of trace fragments of length 1, i.e. the set of individual states.

We can generalise that, in order to answer any question, we need trace fragments of length k where $0 < k \leq t$ and t is the maximum number of time steps in the simulation. The sample space is then defined as the set of all fragments of length k, i.e. we have $\Omega = \mathcal{F}_k$. The σ-algebra Σ is a subset of the power set of Ω and thus represents the set of all possible sets of trace fragments of length k, i.e. $\Sigma \subseteq \mathbb{P}\,\mathcal{F}_k$.

In order to define a probability measure for any event in Σ, we first need to define the probability of a certain trace fragment. The probability of fragment $f = \langle s_j, s_{j+1}, ..., s_k \rangle$ of trace $t = \langle s_0, s_1, ..., s_n \rangle$ where $0 \leq j \leq n$ and $j \leq k \leq n$ is the probability of trace t divided by the number of coherent fragments of t of size $(k - j)$:

$$Pr(\langle s_j, ...s_k \rangle) = \frac{\prod_{0 \leq i < n} P(s_i, s_{i+1})}{n - (k - j) + 1} \tag{3}$$

The probability measure for any event $\sigma \in \Sigma$ (which represents a set of trace fragments) can then be defined as the sum of the probabilities of each trace fragment $\omega \in \sigma$:

$$Pr(\sigma) = \sum_{\omega \in \sigma} Pr(\omega) \tag{4}$$

The association of a probability space with a simulation transition system makes it possible to talk about events and their probability. Events are described by *properties*. A property refers to a set of possible outcomes of a simulation. Consider, for example, a property φ which states that the system will *eventually* reach a given state s. This clearly needs to be answered on *full simulation traces*, i.e. the set of outcomes is defined as the set of all trace fragments of length t where t is the maximum trace length. $\Sigma = \{\sigma \in \mathcal{F}_t \mid \sigma \models \varphi\}$ is then defined as the set of those trace fragments σ that satisfy this condition (denoted $\sigma \models \varphi$) and thus eventually end up in state s. On the other hand, let ψ denote a property that states that the population transitions from state x to state y. This represents a statement about the full population, yet, due to its focus on transitions, it requires trace fragments of length 2 in order to be answered correctly, i.e. we have $\Sigma = \{\sigma \in \mathcal{F}_2 \mid \sigma \models \psi\}$. As a final example, let ψ denote a property which states that *a single agent* transitions from state x to state y. Similar to the previous property, it describes a state transition and thus requires trace fragments of length 2. However, it is also of individual nature, i.e. the set of outcomes that it refers to is the set of all fragments of length 2 *of individual agent traces*. By formulating the properties in the appropriate way, we can answer quantitative properties about the behaviour of

the full population, about the behaviour of groups within the population or about the behaviour of individual agents. By evaluating an individual property on independent and randomly chosen agent traces, we can even answer questions about the *average behaviour* of individual agents. We refer to this process as *in-trace sampling*.

Since, as described above, the traces of a simulation are measurable, the probability of any property ϕ is defined as the sum of the probabilities of all trace fragments of length k in the associated σ-algebra $\Sigma = \{\sigma \in \mathcal{F}_k \mid \sigma \models \phi\}$:

$$Pr(\phi) = \sum_{\sigma \in \Sigma} Pr(\sigma) \tag{5}$$

In order to make clear what fragment size a property is being verified upon (and thus, which interpretation of the sample space is being chosen), we add the sample size as a subscript variable to Pr. For example, we refer to the probability of a property ϕ that is to be evaluated upon trace fragments of size 2 as $Pr_2(\phi)$. We omit the subscript if (i) the formula is to be evaluated upon sets of *full* traces, or, (ii) if the fragment size does not matter for the purpose of description.

This concludes the description of events, properties and their probability in an abstract way. Let us briefly summarise the ideas described above. Essentially, a simulation trace, i.e. the output of a single simulation run, can be seen as a single sample from the set of finite traces defined by the underlying model. Following this interpretation, the set of outcomes, i.e. the set that events and thus also properties are being formulated upon, is fixed as the set of finite traces. However, a single simulation trace can also be interpreted as a set of sample states drawn from the set of states, as a set of sample state tuples drawn from the set of state tuples, as a set of sample state triples drawn from the set of state triples, and so forth. Furthermore, sampling can be performed on the macro, meso and micro level and thus refer to the behaviour of the population, of groups of agents or of individual agents. Depending on how the set of possible outcomes is interpreted, different events can be defined which, ultimately, allows for the expression of richer properties. Given a property ϕ, its meaning and, of course, also its probability may vary depending on which set of outcomes it is interpreted on. It is therefore important to make the fragment size a central parameter of the verification algorithm.

3.3 Complexity

At this point, it is useful to briefly discuss the implications of trace fragmentation on the complexity of verification. Any trace of length t contains $t - k + 1$ coherent fragments of length k. Let c denote the complexity of checking a temporal property in a given language (e.g. LTL) on a trace of length t. Checking the same property on trace *fragments* of length k, increases the complexity to $(t - k + 1) \cdot c$. Fragmentation thus adds a factor that is linear in the length of the trace to the overall complexity of verification.

4 Formulating multiagent correctness properties

In order to be able to exploit the ideas described above and formulate properties about multiagent systems on different observational levels, we need an appropriate specifica-

tion language. For illustration, we define below a simple LTL-based property specification language $\mathcal{L}$ which allows for the formulation of properties about *individual agents* as well as about *arbitrary groups of agents*. $\mathcal{L}$ here is a simplified version of simLTL [5], the specification language used in the verification tool MC^{2}MABS [4]. The syntax of $\mathcal{L}$ is subdivided into two separate layers, an *agent layer* and a *population layer*, which allows for a distinction between *agent properties* ϕ_a and *population properties* ϕ_p. The syntax of agent and population formulae is defined as follows.

$$\phi_a ::= p \mid \mathbf{true} \mid \neg\, \phi_a \mid \mathbf{X}\phi_a \mid \phi_a\, \mathbf{U}\, \phi_a \mid att \bowtie val$$
$$\phi_p ::= p \mid \mathbf{true} \mid \neg\, \phi_p \mid \mathbf{X}\phi_p \mid \phi_p\, \mathbf{U}\, \phi_p \mid att \bowtie val$$

Here, p denotes a Boolean proposition, $att : Name$ denotes an attribute name and $val : Value$ denotes an attribute value (see Section 3), and $\bowtie \in \{=, \neq, <, \leq, >, \geq\}$ is a comparison operator. ϕ_a describes the syntax of an *agent property*, i.e. a property formulated about the behaviour of an individual agent; ϕ_p describes the syntax of a *population property*, i.e. a property formulated about the behaviour of the entire population. The syntax of any formula $\phi \in \mathcal{L}$ is defined as follows:

$$\phi ::= \langle\!\langle \phi_a \rangle\!\rangle \phi_p \mid [ag]\phi_a$$

$\langle\!\langle \phi_a \rangle\!\rangle \phi_p$ describes a *selective population property* which is true if and only if ϕ_p is true for the group of all agents that satisfy property ϕ_a. $[ag]\phi_a$ describes an *indexed agent property*; $ag : Ag$ denotes an identifier that specifies which agent trace within the current simulation trace formula ϕ_a is to be evaluated upon[2].

The ability to formulate properties about groups of agents as well as about individual agents is important, yet there is more to be done. In a multiagent context, we often have to deal with large populations of agents. In this case, in addition to the probability of a property about a *particular* agent, it is also interesting to obtain the probability of a property about the *average* agent. For example, instead of asking for the probability of the income of agent 1 falling below x, we may be interested in the probability of an agent's income falling below x *on average*. This can be achieved through *in-trace sampling*, i.e. the repeated evaluation of an agent property on randomly selected agent traces as described in Section 3. In order to integrate this mechanism into our language, we simply assume that, if the agent identifier is omitted, then the property is checked on a *uniformly randomly chosen agent trace*.

5 Quantitative trace-based analysis

The purpose of this section is to illustrate the usefulness of trace fragmentation in combination with sampling for the purpose of quantitative analysis. In Section 3, the probability of a property has been defined as the sum of the probabilities of all traces (or, more precisely, trace fragments) in the associated σ-algebra. Or, in other words, the probability of ϕ being true in a set of traces Tr denotes the ratio between those traces

[2] For the sake of simplicity, we assume that agents are numbered from 1 to n and that the number of agents is fixed.

$tr \in Tr$ for which ϕ holds (denoted $tr \models \phi$) and those traces $tr' \in Tr$ for which ϕ does not hold (denoted $tr' \not\models \phi$). It remains to discuss, *how* this probability can be computed practically. Clearly, if a complete set of traces is available, then the exact probability can be obtained in a straightforward way, by simply counting those traces for which ϕ holds and dividing their number by the overall number of traces. In general, however, complete sets of traces cannot be assumed to be available. Given the vast size of real-world state spaces, the number of possible traces will be too large and we can only expect to have access to a small subset. In this case, statistical analysis is used to *esimate* the actual probability of a property [11]. In the remainder of this paper, when we refer to a probability $Pr(\phi)$, we thus always mean the estimated probability.

5.1 Analysis types

Up until now, we have completely ignored the fact that properties correspond with trace *fragments* rather than with full traces, as described in Section 3. In this section, we bring together the two ideas of (i) *probability estimation* and (ii) *trace fragmentation* in order to describe advanced types of quantitative analysis. In the following paragraphs, we are mostly interested in the relationship between *states* of a system. States correspond with trace fragments of length 1 which, in turn, correspond with atemporal properties (i.e. properties that do not contain a temporal operator). To that end, we denote with $\mathcal{L}^a$ the *atemporal subset* of $\mathcal{L}$. Furthermore, we abbreviate $(\phi_1 \wedge \mathbf{X}\phi_2)$ with $\phi_1 \rightarrow \phi_2$.

State residency: We start with the notion of a *state residency probability*, i.e. the probability of *being* in a certain state. Informally, of all the time spent in any state, the residency probability of state s describes the fraction of time that is spent in s. Properties about individual states are inherently atemporal in nature and thus correspond with trace fragments of length 1. Given an atemporal property ϕ, the probability of an agent (or any groups of agents) being in a state that satisfies ϕ can then be obtained by simply calculating the probability of ϕ on trace fragments of length 1, i.e. $Pr_1(\phi)$. The *state residency probability* srp can thus be formally defined as follows:

$$srp : \mathcal{L}^a \rightarrow \mathbb{R}$$
$$\forall\, \phi : \mathcal{L}^a \bullet srp(\phi) = Pr_1(\phi)$$

Transition residency: In addition to the probability of being in a certain state, it is crucial to ask properties about the *transitions* between states. Similar to the residency w.r.t. states defined above, we may, for example, be interested in how much of its time a given agent spends in a particular transition. This can be calculated by simply obtaining the probability of a temporal succession property describing on trace fragments of size 2 (because of the 'next' operator). For example, if we are interested in the transition from ϕ_1 to ϕ_2 (where both ϕ_1 and ϕ_2 are atemporal), then the transition residency probability can be obtained by calculating $Pr_2(\phi_1 \rightarrow \phi_2)$. This leads to the following formal description of the transition residency probability trp:

$$trp : \mathcal{L}^a \times \mathcal{L}^a \rightarrow \mathbb{R}$$
$$\forall\, \phi_1, \phi_2 : \mathcal{L}^a \bullet trp(\phi_1, \phi_2) = Pr_2(\phi_1 \rightarrow \phi_2)$$

Transition probability: The purpose of the next type of quantitative analysis is to determine a particular transition probability, i.e. the probability of transitioning into a particular successor state in which ϕ_2 holds, given that we are currently in state in which ϕ_1 holds. The transition probability is obtained by dividing the probability of transitioning from ϕ_1 to ϕ_2 by the residency probability of ϕ_1, i.e. $trp(\phi_1, \phi_2)/Pr_2(\phi_1)$. It is important to note that both probabilities need to be obtained on trace fragments of size 2 (even the second, atemporal one!). A formal definition of the transition probability tp can now be given as follows:

$$tp : \mathcal{L}^a \times \mathcal{L}^a \to \mathbb{R}$$
$$\forall\, \phi_1, \phi_2 : \mathcal{L}^a \bullet tp(\phi_1, \phi_2) = trp(\phi_1, \phi_2)/Pr_2(\phi_1)$$

Correlation analysis: Probabilistic analysis can be used conveniently to determine *probabilistic dependence* or *correlation*. Correlation analysis represents an important building block in the quality assurance process. It can give insights into the system's dynamics by revealing behaviours that are coupled, i.e. whose occurrence is (entirely or to some extent) synchronised. Furthermore, the analysis of correlations may indicate *causal relationships* and can thus be used to detect symptoms that can motivate further, more tailored experiments. For example, if A and B are positively correlated, one can be sure that one of the following three facts is definitely true: (i) A is a cause of B, (ii) B is a cause of A or, (iii) there is a common cause for A and B. Positive correlation can be defined formally as follows[3]:

$$posCorr : \mathcal{L} \times \mathcal{L} \to \{\mathbf{true}, \mathbf{false}\}$$
$$\forall\, \phi_1, \phi_2 : \mathcal{L} \bullet posCorr(\phi_1, \phi_2) \Leftrightarrow Pr(\phi_1 \wedge \phi_2) > Pr(\phi_1) \cdot Pr(\phi_2)$$

This concludes the description of our analyses. As illustrated in the case study in the next section, quantitative analysis becomes most powerful if it is performed on different observational levels.

6 Case study

In order to illustrate the usefulness of quantitative analysis in the context of trace-based verification, we introduce a small example scenario from the area of swarm robotics. The choice is motivated by the fact that, albeit often conceptually startlingly simple, swarm models exhibit a significant level of complexity which typically prevents them from being amenable to conventional formal verification. On the other hand, they may require a high level of provable correctness. We show how, through statistical model checking in combination with quantitative analysis as described above, interesting properties which reach beyond pure reachability and safety checking can be answered efficiently and with a good level of precision. We focus here on *foraging*, a problem which has been widely discussed in the literature on *cooperative robotics* [2]. Foraging describes the process of a group of robots searching for food items, each of which

[3] The definition of functions for negative correlation and non-correlation, i.e. statistical independence, are omitted; they can be given accordingly.

delivers energy. Individual robots strive to minimise their energy consumption whilst searching in order to maximise the overall energy intake. The study of foraging is important because it represents a general metaphor to describe a broad range of (often critical) collaborative tasks such as *waste retrieval*, *harvesting* or *search-and-rescue*. A good overview of multirobot foraging has been given by Cao *et al.* [2].

All experiments described below were conducted on a Viglen Genie Desktop PC with four Intel® Core™ i5 CPUs (3.2 GHz each), 3.7 GB of memory and Gentoo Linux (kernel version 3.10.25) as operating system, using the verification tool MC^{2}MABS [4]. Results are based on experiments involving 100 replications of the given model.

Model description: The model described in this section is based on the work of Liu *et al.* [12]. In the model, a certain number of food items are scattered across a two-dimensional space. Robots move through the space and search for food items. Once an item has been detected within the robot's field of vision, it is brought back to the nest and deposited which delivers a certain amount of energy to the robot. Each action that the robot performs also consumes a certain amount of energy. The model is deliberately kept simple. Each robot can be in one of five states: *searching* for food in the space, *grabbing* a food item that has been found, *homing* in order to bring a food item back to the nest, *depositing* a food item in the nest, and *resting* in order to save energy. Transitions between states are probabilistic and either fixed or (which is clearly more realistic) dependent upon the state of the other agents, as described in the following section. The overall swarm energy is the sum of the individual energy levels. Furthermore, in order to make things more interesting, we assume that there are initially two different types or *makes* of agent which only differ in terms of their field of vision.

Instead of viewing a population of robots as an abstract entity in which agents have a certain probability of finding food (as, for example, done in [8]), we focus here on an agent-based representation of the scenario in which the world that robots inhabit is represented explicitly. It is common to many agent-based models to model the environment as a two-dimensional grid, in our case a grid of 100×100 cells. Each grid cell can be inhabited by an arbitrary number of agents. Food items are distributed uniformly across the grid. In the current version of the model, there are 1,000 food items distributed across 10,000 grid cells, which amounts to a food density of 10%. Agents of make 0 are able to detect all food items within a radius of 1, agents of make 1 are able to detect all food items within a radius of 4. The behavioural protocol that each agent follows is shown below.

- If $searching$: look for food. If food has been found, move to the cell and start $grabbing$; otherwise remain $searching$. If no food can be found within T_s time steps, start $homing$.
- If $grabbing$: if the food is still there after T_g time steps, grab it and start $depositing$; otherwise start $homing$.
- If $depositing$: start $resting$ after T_d time steps.
- If $homing$: start $resting$ after T_h time steps.
- If $resting$: start $searching$ after T_r time steps.

It is important to stress that our goal is not to construct an overly realistic model here; the main focus is on illustration and the model is thus kept deliberately simple.

Table 1. Transition probabilities for all agents, agents of make 0 and agents of make 1

	All agents		Make 0		Make 1	
Transition	Total time	Probability	Total time	Probability	Total time	Probability
$S \to G$	1m 52s	0.1701	2m 40s	0.0363	2m 50s	0.6223
$G \to D$	2m 01s	0.1735	2m 40s	0.1945	2m 40s	0.1678
$G \to H$	2m 09s	0.0076	2m 46s	0.0034	2m 42s	0.0090
$D \to R$	2m 02s	0.1868	2m 50s	0.1912	2m 42s	0.2111
$R \to S$	2m 03s	0.1913	2m 55s	0.1950	2m 47s	0.1833

Despite its conceptual simplicity, however, the model already exhibits a significant level of complexity which prevents it from being amenable to conventional exhaustive verification. Due to the use of floating point variables for the agents' energy levels, the state space is effectively infinite. Even if limited to a comparatively small value (e.g. 1,000), the number of possible states would be beyond what is currently verifiable formally.

Verification: Our goal is to determine whether the model is reasonably robust, i.e. whether agents have enough energy during the simulated timespan. We start with the following (largely arbitrary) parametrisation:

- 100 agents, 1,000 ticks
- Time spent in each state: $T_s = T_g = T_r = T_h = T_d = 5$
- Energy consumed in each state: $E_s = 12, E_g = 12, E_h = 6, E_r = 2, E_d = 62$
- Initial level of energy per agent: 40

In order to check whether the parametrisation shown above already satisfies the given requirements, we first define the following population-level property which states that the swarm as a whole will never run out of energy (note the use of '$\langle\!\langle \mathbf{true} \rangle\!\rangle$' to refer to the whole population):

$$\phi_1 = \mathbf{G}(\langle\!\langle \mathbf{true} \rangle\!\rangle (swarm\_energy \geq 0))$$

Despite every robot having 40 units of initial energy, the verification of Property ϕ_1 returns a probability of 0 which shows that the parametrisation given above is not suitable for this version of the model. In order to gain a deeper understanding of why this may be the case, it is useful to study how frequently robots switch from one state into another by determining their *average transition probabilities*. Following the description in Section 5, this requires the comparison of different probabilities, each of which has been obtained on trace fragments of length 2. We illustrate the formulation for the transition probability from searching to grabbing. In order to verify this property, we need the following two subformulae: $\phi_G = grabbing$ and $\phi_S = searching$. The overall transition probability for an individual agent is then calculated as follows:

$$Pr(S \to G) = tp(\phi_S, \phi_G) \tag{6}$$

The results for 100 replications are shown in the first section of Table 1[4]. We can see that robots have an equal probability of finding and grabbing food ($\approx 17\%$). We can

[4] For clarity, we abbreviate states with their capitalised first letters in all subsequent tables.

also see that agents have a very low probability of transitioning into the homing state, which is positive since homing is always caused by a timeout and is thus undesirable.

However, when calculating the transition probabilities, we need to take into account that we have two different makes of agent, each of which can be expected to have different probabilities. In order assess whether this is really the case, we could, for example, check whether being of make 0 is *positively correlated* (or being of make 1 is negative correlated, respectively) with finding food. We will instead 'zoom in' and assess robots of different makes separately. This can be achieved by using a selection operator $\langle\langle\rangle\rangle$ as described in Section 4. For example, for robots of make 0 (for which we assume that proposition $make0$ is always true), the properties necessary for calculating the transition probability $Pr(S \rightarrow G)$ from searching to grabbing can be formulated as follows: $\psi_S = \langle\langle make0 \rangle\rangle grabbing$ and $\psi_G = \langle\langle make0 \rangle\rangle searching$. The overall transition probability can then be calculated similar to the previous property, i.e. $Pr(S \rightarrow G) = tp(\psi_S, \psi_G)$. The results for all checks are shown in Table 1. It is obvious that robots of make 1 have a significantly higher probability of finding food which, given their larger field of vision, is intuitively correct. What is also interesting, however, is that a robot's make seems to have a small but obvious impact on its probability of grabbing food; this is indicated by the lower probability of transitioning from grabbing to depositing for robots of make 1. One possible explanation is that, due to their larger field of vision and their consequently higher probability of finding food, robots of make 1 may block each other by 'stealing' food that is already aimed for by a different robot. This explanation may also be underpinned by the slightly higher probability of robots of make 1 moving from grabbing to homing than robots of make 0: the only reason for performing this transition is that a food item aimed for is lost to a different agent. Given the small sample size, however, care needs to be taken when interpreting the numbers — especially when differences are very small, as in this case.

Table 2. Expected individual transition prob. and prob. of constant positive swarm energy

Vision	$Pr(S \rightarrow G)$	$Pr(G \rightarrow D)$	$Pr_t(\phi_1)$	$trp(depositing, resting)$
1	0.3416	0.1889	0.0	0.0108
2	0.1344	0.1853	0.0	0.0300
3	0.3735	0.1711	0.0	0.0438
4	0.6571	0.1631	0.0	0.0460
5	0.8364	0.1630	0.0	0.0470

The numbers seem to suggest that the size of the field of vision has a positive impact on the food finding probability and a slightly negative impact on the food grabbing probability. This hypothesis can be investigated further by performing a range of experiments in which the vision parameter is constantly increased. The results are shown in Table 2. The numbers in the second column indicate that, in fact, the size of the field of vision has a significant positive impact on the probability of finding food (as expected). This shows that there is a *causal dependence* between an agent's field of vision and its probability of finding food. The numbers in the third column indicate that there is a slightly negative correlation between the field of vision and the probability of grabbing

food. The fourth column of the table shows the probability of Property ϕ_1 which is 0 in all cases; varying the field of vision alone is thus not sufficient for sustaining a positive energy level (at least not in the current scenario).

The numbers suggest that, despite the slight loss in grabbing probability, the swarm designer is best off by giving all robots a high field of vision. In order to confirm this assumption, we can formulate another property which denotes the *overall probability of an agent gaining energy*. Remember that energy is always gained in the final time step of the depositing state, i.e. before the agent starts resting. In order to determine the overall probability of an agent gaining energy, we can formulate the following property:

$$trp(depositing, resting) = Pr_2(depositing \rightarrow resting) \tag{7}$$

Since this is a property whose truth needs to be ascertained on state transitions, it needs to be checked on trace fragments of size 2. It is also important to note that, since it is not conditional upon the agent's being depositing (i.e. it does not use logical implication), this property does *not* describe a transition probability in its strict sense. Instead, it describes the *overall* probability of performing this particular transition and can thus be used to determine the overall probability of an agent gaining energy. For simplicity, we assume that 5 is the maximum level of vision that can be realised technically. The verification results are shown in the last column of Table 2. They strengthen the assumption that the scenario with the largest field of vision is the most efficient one since, in this case, agents are most likely to gain energy.

Table 3. Expected individual state distribution

Vision	$srp(searching)$	$srp(grabbing)$	$srp(homing)$	$srp(resting)$	$srp(depositing)$
1	0.2952	0.0563	0.2694	0.3234	0.0555
2	0.2147	0.1553	0.1668	0.3122	0.1505
3	0.1251	0.2486	0.0810	0.3119	0.2337
4	0.0873	0.2896	0.0470	0.3083	0.2647
5	0.0691	0.3087	0.0330	0.3109	0.2763

The numbers so far give a strong indication that the probability of grabbing food should be increased. In order to choose the right strategy for achieving this goal, it is essential to *explain* its current level first, i.e. to understand *why* it is so low. The intuitive assumption is that an increased field of vision also increases competition among robots which itself increases the probability of agents missing out when trying to grab food. This assumption can be checked by determining the *expected state distribution*, i.e. the *amount of time a robot is expected to spend in each of the states*. The properties are formulated with the help of the state residency probability srp described in Section 5.1. The expected state distribution can be obtained by checking all properties above on trace fragments of size 1, i.e. on individual states. This is important since the properties are state properties and, in order to determine their probability, we thus need to sample from the distribution of states. The verification results are shown in Table 3 (the Roman literals denote the individual simLTL properties described in this chapter). It becomes apparent that in case of lower vision, a significantly higher proportion of robots

Table 4. Expected state distribution and energy development for $T_r = 1$ and $T_g = 1$

Vision	Scenario	$srp(s)$	$srp(g)$	$srp(h)$	$srp(r)$	$srp(d)$	$Pr_t(\phi_1)$	$Pr_t([](energy > 0))$
5	$T_r = 1$	0.0925	0.4095	0.0466	0.0828	0.3691	0.0	0.0
5	$T_g = 1$	0.0923	0.0816	0.0262	0.4135	0.3893	1.0	0.78

spend their time searching and homing (due to timeouts) than in case of higher vision. However, it also becomes apparent, that in case of higher vision, a significantly higher proportion of agents spend their time grabbing. This suggests that grabbing becomes a bottleneck which impedes foraging. Apart from grabbing, in all scenarios, a significant number of agents spend their time resting.

We now have two possible directions to improve the overall efficiency of the swarm: we can either try to decrease the time individuals spend for resting or we can try to decrease the time spent for grabbing food items. In order to compare the effect of both changes, we determine again the expected state distribution for each of the two cases. The results are shown in Table 4[5]. Reducing the resting time to 1 has the effect of forcing more robots into searching, grabbing and depositing. Likewise, reducing the grabbing time to 1 forces more robots into searching, resting and depositing. Both scenarios only differ with respect to the number of agents grabbing or resting. Taking into account the energy consumption of each agent intuitively suggests that scenario 2 (reduced grabbing time) must be significantly more effective since, in this case, more agents are resting and resting consumes significantly less energy than depositing. This assumption can be strengthened by looking at the overall probability of Property ϕ_1 (shown in Column 7) of Table 4. In the case of reduced resting time, the probability of the swarm always having positive energy is 0; in the case of reduced grabbing time, the probability is 1.0. In terms of individual energy levels, individual robots have an *average probability* of always having positive energy of $\approx 78\%$, as shown by the unindexed individual agent property in Column 8 of Table 4.

We have now reached a situation in which the overall swarm energy level as well as the majority of all individual energy levels are always positive. This concludes our small case study. In fact, there is still a significant number of individual robots ($\approx 22\%$) running out of energy. Their calibration, however, is not further discussed here.

7 Conclusions and future work

Statistical model checking can provide a powerful alternative for the verification of systems that are unamenable to conventional formal verification. Because of its focus on finite traces, statistical verification is typically focussed on comparatively simple properties. This critically limits the verifiability of large-scale multiagent systems with their complex, internal structure. In this paper, we showed how, by combining statistical verification with an advanced type of sampling and trace fragmentation, interesting quantitative analyses on different observational levels can be performed. Using a simple case study from the area of swarm robotics, we showed that, albeit approximate in

[5] For space limitation, the states are abbreviated with lower-case letters, e.g. s for $searching$.

nature, those types of analyses can be helpful to shed light on the dynamics of complex systems and uncover some of their internal mechanisms.

In this paper, we restricted our attention to a small number of quantitative analyses. Combining the expressiveness of temporal logics with statistical verification, a much wider range of analyses is possible. For example, in statistical time series analysis, correlation can be generalised to the temporal case by measuring the *autocorrelation* of a time series. The same idea could be applied in a trace-based verification scenario. Furthermore, probabilistic analysis provides an interesting basis for the analysis of *causal* relationships, either in a statistical sense (e.g. *Granger causality*) or by utilising probabilistic theories of causation [6]. In a verification context, causal analysis is a powerful tool for the *explanation* of phenomena. We plan to further investigate this idea, with a particular focus on the work of Kleinberg and Mishra [7].

References

1. C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
2. Y. U. Cao, A. S. Fukunaga, and A. Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, Mar. 1997.
3. B. Finkbeiner, S. Sankaranarayanan, and H. B. Sipma. Collecting statistics over runtime executions. *Formal Methods in Systems Design*, 27(3):253–274, November 2005.
4. B. Herd. *Statistical runtime verification of agent-based simulations*. PhD thesis, King's College London, 2015.
5. B. Herd, S. Miles, P. McBurney, and M. Luck. An LTL-based property specification language for agent-based simulation traces. Technical Report 14-02, King's College London, Oct 2014.
6. C. Hitchcock. Probabilistic causation. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter 201 edition, 2012.
7. S. Kleinberg and B. Mishra. The temporal logic of causal structures. In *Proc. 25th Conf. on Uncertainty in Artificial Intelligence*, pages 303–312. AUAI Press, 2009.
8. 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 *LNCS*, pages 440–447. Springer, 2010.
9. M. Kwiatkowska, G. Norman, and D. Parker. Quantitative analysis with the probabilistic model checker PRISM. *Electronic Notes in Theoretical Computer Science*, 153(2):5 – 31, 2006. Proc. 3rd Workshop on Quantitative Aspects of Programming Languages.
10. M. Kwiatkowska, G. Norman, and D. Parker. Prism 4.0: Verification of probabilistic real-time systems. In *Proc. 23rd Int. Conf. on Computer Aided Verification*, pages 585–591. Springer, 2011.
11. A. Legay, B. Delahaye, and S. Bensalem. Statistical model checking: an overview. In H. Barringer, Y. Falcone, G. Roşu, B. Finkbeiner, O. Sokolsky, K. Havelund, I. Lee, G. Pace, and N. Tillmann, editors, *Proc. 1st Int. Conf. on Runtime Verif.*, pages 122–135. Springer, 2010.
12. W. Liu, A. Winfield, and J. Sa. Modelling swarm robotic systems: A case study in collective foraging. In M. S. Wilson, F. Labrosse, U. Nehmzow, C. Melhuish, and M. Witkowski, editors, *Towards Autonomous Robotic Systems*, pages 25–32, 2007.
13. V. Nimal. Statistical approaches for probabilistic model checking. MSc Mini-project Dissertation, Oxford University Computing Laboratory, 2010.
14. U. Sammapun, I. Lee, O. Sokolsky, and J. Regehr. Statistical runtime checking of probabilistic properties. In *Proc. 7th Int. Conf. on Runtime Verif.*, pages 164–175. Springer, 2007.