Statistical Model Checking of Python Agent-Based Models: An Integration of MultiVeStA and Mesa

Andrea Vandin[1,2(✉)]

[1] Sant'Anna School of Advanced Studies, Pisa, Italy
andrea.vandin@santannapisa.it
[2] DTU Technical University of Denmark, Lyngby, Denmark

Abstract. Agent-based models (ABM) consist of several heterogeneous agents interacting in a complex way, possibly mediated by spatial constraints or other aspects, giving rise to emergent behavior not directly expressed in the model itself. These aspects made ABMs widespread in several areas, including the social sciences. These models are typically too complex to be solved analytically, requiring to use simulation-based analyses. Often, especially in the social sciences, these simulation-based analyses are not automatic, and the number of performed simulations or simulation steps might be arbitrary. This might lead to replicability issues, to low statistical accuracy of the results, or just to wrong results. In computer science, simulation-based analyses can be automated, e.g., using statistical model checking. We present an integration of the statistical model checker MultiVeStA with Mesa, a python-based framework for modeling and analysing ABMs. We validate the integration by using two seminal ABMs from the social sciences. We analyze the famous Boids flock model, able to generate flocking behaviors of birds, with the transient and counterfactual analysis capabilities of MultiVeStA. We analyze the well-known Schelling model, able to generate social segregation behaviors, with the steady-state and ergodicity diagnostics capabilities of MultiVeStA. The contribution of this paper is not methodological. Rather, this is on the one hand a case study paper presenting an application of MultiVeStA. On the other hand, it is a step towards automating and making more reliable the simulation-based analysis of models written in the popular Mesa framework.

Keywords: Statistical Model Checking · Agent-based model · MultiVeStA · Mesa · Transient analysis · Steady-state analysis

1 Introduction

Agent-based Models (ABMs) are characterized by complex interactions of relatively simple heterogeneous agents, leading to emergent behaviors that are not explicitly defined within the model. The use of ABMs has spurred in several

B. Steffen (Ed.): AISoLA 2024, LNCS 15217, pp. 398–419, 2025.
https://doi.org/10.1007/978-3-031-75434-0_26

disciplines like, e.g., ecology [37], health care [30], geography [15], and medicine [6]. ABMs are becoming more and more popular also in the social sciences (see, e.g., [27,29,31,33,50,56,59,71]), where they serve as a pivotal tool for analyzing by simulation phenomena that are too intricate to be addressed analytically.

The simulation of ABMs involves the iterative execution of each agent's step. Depending on the analysis of interest, the design of the simulation experiments might play a crucial role in the reliability of the obtained results [67]. Moreover, the analyses, especially in the social sciences, lack automation, which can result in issues in replicability, statistical inaccuracy, or erroneous results [14,67,76].

Statistical model checking (SMC) is a family of simulation-based analysis techniques from computer science which aims at automating simulation experiments, freeing the modeler from the burden of designing simulation experiments and testing the reliability of the obtained results [2,47]. So-called black-box SMC does not make any assumption on the model under analysis, except that it can be simulated probabilistically [68,81], making it general and easily applicable to existing simulation models. MultiVeStA [35,66,76] is a black-box SMC tool that can be easily integrated with existing simulation models. MultiVeStA has been recently redesigned and extended to target ABMs from the social sciences, and in particular from economics [76]. It has been shown that it can automate and speed up analyses of interest in the community, and avoid erroneous analysis spurring from wrong designs of simulation experiments [76]

In computer science, several frameworks and formalisms have been proposed to model and analyze ABMs (see, e.g., [13,38–40,43,54,55,60–62,72]). However, ABM models from the social sciences are still typically written from scratch using general-purpose programming languages like Java [17], or C [28]. Some ABM frameworks have emerged over the years in the social sciences, like, e.g., Netlogo [79], LSD [73], jasmine [64], or Mesa [41]. In this paper, we present an integration of MultiVeStA with Mesa, a Python-based framework for ABM modeling and analysis popular in the social sciences [41]. This makes MultiVeStA and SMC available for analyzing all models available in the Mesa model repository. These include popular ABM models from the social sciences like the Boids flocks model [63], which studies flocks behaviors of birds, and the Schelling model [65] which studies phenomena of social segregation of minority groups. We validate our integration using these two seminal ABMs from the social sciences. We employ MultiVeStA's transient (*what is the expected value of a property at given steps?*) and counterfactual (*does a reparameterization have significant behavioral impacts?*) analysis capabilities to study the Boids model, and its steady-state (*what is the expected value of a property at equilibrium?*) and ergodicity diagnostics (*does the system actually have one equilibrium?*) capabilities to analyze the Schelling model. Our work automates the analysis of ABMs written in Mesa, providing insights into the emergent behaviors resulting from the interactions of their agents. By leveraging the statistical model checking techniques, we offer a robust framework for researchers to explore and understand the dynamics of ABMs in social sciences.

Notably, the scope of this paper is neither that of discussing the peculiarities of ABMs, nor whether they could be modeled with the very popular and powerful tools from the formal verification community (e.g. [11,24,26,44]). Rather, starting from the fact that a wide research community creates and uses ABMs, and that Mesa is gaining popularity in this area, this paper proposes improvements to the analysis of ABMs written in Mesa. Nevertheless, this can be seen as a further contribution from the formal verification community to the ABM one. On the one hand, this is a case study paper presenting an application of MultiVeStA's extensions from [76] to different domains. On the other hand, this paper represents a step towards automating and making more reliable the simulation-based analysis of ABM Mesa models.

Synopsis. The paper is structured as follows: Sect. 2 introduces SMC and MultiVeStA, Sect. 3 introduces Mesa, while Sect. 4 discusses their integration. Sections 5 and 6 present the Boids flocks model and the Schelling one, respectively, and their analyses with MultiVeStA. Section 7 concludes the paper.

2 Statistical Model Checking and MultiVeStA

We overview statistical model checking techniques available in MultiVeStA.

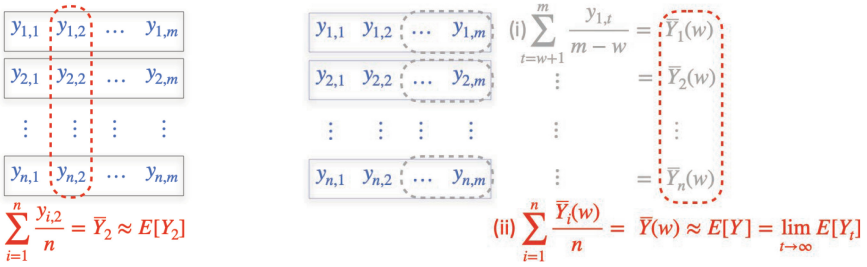
2.1 Overview

Statistical model checking (SMC) entails conducting a sufficient number of probabilistic simulations of a model, each of a sufficient number of simulation steps, to derive statistically-reliable estimations of its properties [2,47]. Black-box SMC is a variant of SMC where no assumption is made regarding the studied model, except that it supports probabilistic simulations [68,81]. This has the drawback of making more difficult the use of advanced techniques, like, e.g., those to handle rare events [48] or those based on machine learning to reduce the number of simulations [12]. On the other hand, this makes black-box SMC very general and virtually applicable to any simulation model.

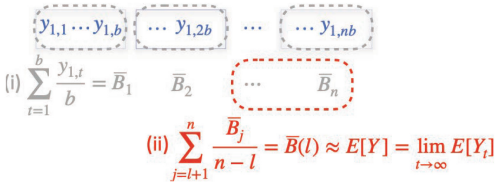
Here we consider MultiVeStA [35,66,76], a statistical analyzer that can be tool-chained with existing simulators, enriching them with black-box SMC capabilities. The tool-chaining only regards basic functionalities supported by any discrete-event simulator. In particular, MultiVeStA only needs to be instructed on how to *reset* the simulator before running a new simulation; perform *one step* of simulation; *evaluate* an observation in the current simulation state. This allowed to integrate MultiVeStA with many simulators on several domains, including crowd steering scenarios [58], public transportation systems [21,36], lending pools in decentralized finance [7], highly-configurable systems [9,74], business process modeling [23], security threat modeling [8,19], adaptive robotic systems [10,16], and collective adaptive systems in general [34].

Originally [66], MultiVeStA supported only transient properties (evaluated at given points in time or upon a given condition), like the expected value of the number of people who successfully escaped a critical situation within

10 min [58]. Later [35], it has been extended with limited support for steady-state properties, like the expected concentration of chemical species on the long run [35]. More recently, MultiVeStA has been extended and redesigned to target agent-based models (ABM) from the social sciences, and in particular economic ABMs [75,76]. In this redesign, transient analysis capabilities have been extended with *counterfactual analysis*, crucial in economic ABMs to study whether a change in the parameters (e.g., higher taxation) leads to results significantly different from a statistical point of view (e.g., higher GDP) [76]. However, the emphasis of the extension has been on steady-state analysis, of particular interest for the community. MultiVeStA now features two complementary algorithms for steady-state analysis which extend and combine ideas from state-of-the-art approaches [35,69,70], as well as a methodology for ergodicity diagnostics which can inform the modeler on whether the model allows for steady-state analysis. Replicating the material from [76] is out of the scope of this paper. However, in the rest of this section we try to convey the intuition behind the proposals in [76]. Finally, in a different line of research [18–20], MultiVeStA has been integrated with process-oriented data science techniques (process mining [1]) to validate and debug models by graphically representing the behavior that led to given SMC results.



(a) Transient analysis (b) Steady-state analysis by *Replication and Deletion* (RD)



(c) Steady-state analysis by *Batch Means* (BM) using one long simulation

Fig. 1. Representation of transient and steady-state analysis in MultiVeStA.

2.2 Transient and Steady-State Analysis in MultiVeStA

We informally introduce MultiVeStA's algorithms for transient and steady-state analysis, while we refer to [76] for a more formal and detailed presentation. We

can think of the output of a simulation model as a discrete-time[1] stochastic process $(\mathbf{Y}_t)_{t>0}$ describing the evolution over (simulated) time of a number of variables of interest (e.g., the number of people who successfully escaped a critical situation at time –or step– t, or a 1-0 value denoting whether an event happened or not at time t). For the sake of presentation, let us assume that $(\mathbf{Y}_t)_{t>0}$ contains only one variable of interest $(Y_t)_{t>0}$. Figure 1(a) depicts n independent simulations of a model (one per row), each containing the realization of Y_t in each step $t = 1, \ldots, m$ (one per column). Considering simulation i, i.e., row i, its outcome can be represented as a sequence of observations (or realizations) $y_{i,t} \in \mathbb{R}$. Clearly, the observations within the same row i are not independent, while those in the same column t are independent and identically distributed (IID). On these simulations, we can study two types of properties:

- The red text in Fig. 1(a) exemplifies the analysis of a *transient property* evaluated at the specific time point 2: $E[Y_2]$;
- The red text in Fig. 1(b)–(c) exemplifies the analysis of a *steady-state property* evaluated when the system is at equilibrium: $E[Y] = \lim_{t\to\infty} E[Y_t]$.

The algorithm for transient analysis in Fig. 1(a) to compute a statistical estimate $\overline{Y}_2$ of $E[Y_2]$ is simple: it performs n simulations, collects from each the realization of a property $(y_{i,2})$, and computes their mean. Figure 1(b) depicts an algorithm for steady-state analysis known as Replication and Deletion (RD) [46]. It is similar to the previous one, with an additional pre-processing shown in gray in Fig. 1(b) to compute the *horizontal mean* $\overline{Y}_i(w)$ of each simulation i. Such means are all IID because they come from IID simulations, therefore, as shown in red, we can treat them like the observations $y_{i,2}$ in the transient analysis, computing their *vertical* mean $\overline{Y}(w)$. The horizontal means are not computed on all observations: the initial w ones are discarded. This is the so-called *warmup period* which contains bias due to the initial conditions. It must be identified and discarded to avoid potentially important errors (see, e.g., Sect. 6 of [76]). This is a crucial problem, which can hinder a fully automatic analysis of steady-state properties. In [76], we provided and related to the literature a fully automated algorithm, `autoRD`, which applies involved statistical tests to identify such warmup period, and then performs RD on long-enough simulation steps after its end. Figure 1(c) depicts a complementary algorithm for steady-state analysis based on the notion of Batch Means (BM) [5,22,45]. Differently from RD, which performs *many short* simulations, BM performs *one long* simulation. Such long run is evenly divided into *batches* (adjacent non-overlapping subsamples), and the means within each batch, i.e., the *batch means* $\overline{B}_j$ in Fig. 1(c), are computed. Intuitively, if the studied process satisfies certain statistical properties, and if the simulation run is long enough, each batch mean can be used similarly to an horizontal mean in RD. In [76] we provided and related to the literature a fully automated algorithm, `autoBM`, based on BM.

[1] To simplify presentation, we focus on discrete-time simulation models. However, with mild modifications the algorithms can be adapted to the continuous-time case. Essentially, in that case the focus moves from simulation steps to simulated time.

2.3 Confidence Intervals and Counterfactual Analysis

Figure 1 (a)–(c) depict how to compute statistical estimates for $E[Y_2]$ (i.e., $\overline{Y}_2$) and $E[Y]$ (i.e., $\overline{Y}(w)$, and $\overline{B}(l)$). We enrich such estimates with appropriate measures of uncertainty, α-δ confidence intervals (CI). Given two parameters $\alpha \in (0,1)$ and $\delta \in \mathbb{R}^+$, we guarantee with statistical confidence $(1-\alpha)\cdot 100\%$ that the expected value belongs to the interval of width δ centered at its estimate. This is based on standard statistical techniques supported by the law of large numbers (see, e.g., Chap. 9 of [46]). Intuitively, one just needs to increase the number of considered simulations based on the sample variance of the estimates.

A common exercise when analysing simulation models in the social sciences is counterfactual analysis (see, e.g., [67]). This builds on transient analysis, and consists in comparing the estimates from different model parameterizations to study whether the obtained results differ significantly from a statistical point of view. To answer this, MultiVeStA performs the *Welch's t-test* of equality of the means [77] or the *u-test* [51] on the pair of estimations of every step t of interest. Among the tools that support statistical model checking we mention here PRISM [44]. PRISM offers a so-called "experiment functionality"[2] which allows to run analyses for different predefined parameters and store the results in CSV files. However, to the best of our knowledge, no counterfactual analysis based on statistical tests is then offered for the obtained results.

2.4 Ergodicity Diagnostics

The RD and BM approaches are complementary [4,42,78]. In both cases, a steady-state analysis is meaningful only "around" a statistical equilibrium, requiring that $\lim_{t\to\infty} E[Y_t]$ actually exists and is finite. Therefore, not all properties can be studied at steady-state. In [76], we presented a methodology based on `autoRD` and `autoBM` for ergodicity diagnostics. It assesses whether assumptions necessary for steady-state analysis are clearly violated when analysing a given property on a given model. Our methodology can help in understanding whether the modeler should instead consider a transient analysis for the considered property. It is important to stress that it does not make sense to talk about steady-state analysis or ergodicity diagnostics for *a model*. Rather, these considerations shall be done for a property evaluated on a model. In fact, given a model, certain properties may allow or not for steady-state analysis. Consider, for example, steady-state analyses of actual values (e.g., the GDP of a country) and of their increment rates (e.g., the increment rate of the GDP of a country). If the increment rates eventually reach an equilibrium, then steady-state analyses of increment rates are likely possible, while steady-state analyses of the actual values will not be possible because the actual raw value will keep increasing.

Our methodology can be summarized as:

- Run both `autoRD` and `autoBM` on the considered property. If one fails, or if the results differ significantly (i.e., are further away than the δ of the CI from Sect. 2.3 used for the analysis), a violation has been observed;

[2] https://www.prismmodelchecker.org/manual/RunningPRISM/Experiments.

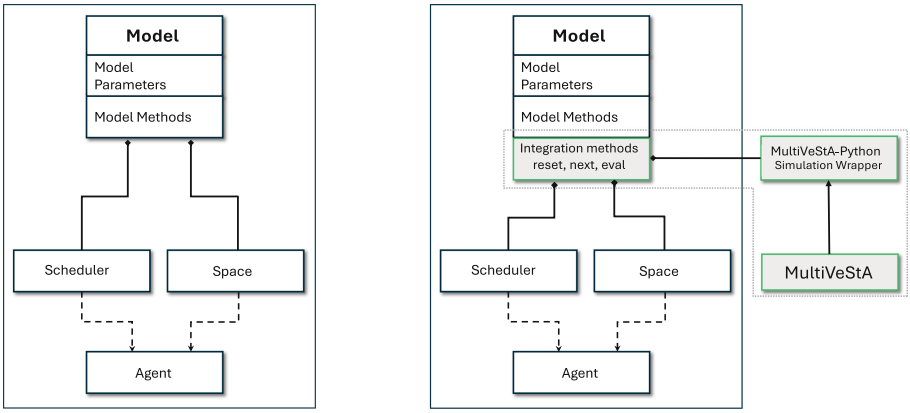


Fig. 2. Simplified structure of a Mesa model (left, adapted from [41]), and its extension to support the integration with MultiVeStA (right).

– Otherwise, check whether the horizontal means of `autoRD` pass a normality test. If this is not the case, a violation has been observed;
– No violation observed otherwise.

3 The Agent-Based Modeling Framework Mesa

An agent-based model (ABM) consists of several heterogeneous agents interacting in a complex way, possibly mediated by interaction constraints, giving rise to *emergent behavior*. For example, in a macroeconomic ABM we can have thousands of households, firms, and banks which interact (working and buying goods, producing goods and paying salaries, lending money, respectively), generating as emergent behavior an economy on which one can estimate macroeconomics indicators like GDP, without them being directly encoded in the model [17]. These models are typically too complex to be solved analytically, requiring to resort to simulation-based analyses. In studies conducted in the social sciences, these simulation-based analyses may be not automated, possibly leading to issues related to replicability of results, to low statistical accuracy, or to wrong results (see, e.g., [67,76]). Simulating an ABM means iterating through each agent, every time in a different order, to trigger single steps of execution of each. An ABM simulation can be seen as a form of discrete-event simulation (DES) [46].

Often, ABMs from the social sciences are written directly in general-purpose programming languages (see, e.g., [17,28,32]). Nevertheless, several frameworks supporting ABM modeling and analysis in the social sciences emerged over the years, like, e.g., Netlogo [79], LSD [73], jasmine [64], or Mesa [41]. Here, we consider Mesa (https://mesa.readthedocs.io/en/stable/), a popular python-based framework for ABMs. Mesa has a modular architecture [41]; Fig. 2 (left) depicts the minimal components required to create a model in Mesa, by extending base

classes from the library. These include a `Model` class, which stores parameters and agents, and triggers their execution; `Agent` classes that define agent characteristics; a `Scheduler` that governs agent activation patterns and manages temporal aspects; and a `Space` class modeling the space where agents interact [41].

4 Integration of Mesa with MultiVeStA

The integration of Mesa with MultiVeStA required us to extend the architecture of Mesa as shown in Fig. 2 (right), making it straightforward to integrate Mesa models with MultiVeStA, enriching them with SMC capabilities.[3] Given a Mesa model, the modeler needs to implement a new method `set_simulator_for_new_simulation` in the class `Model`. This will be invoked by MultiVeStA to reset the model before every new simulation, also providing the random seed to be used in the simulation. Typically, the modeler has to move here part of the code from the `Model` constructor, leaving there only code that can be executed only once and not before every simulation. For example, code to create or reset the `Scheduler`, which is responsible for controlling when the agents are executed, shall be moved in the new method. Furthermore, the modeler typically also needs to move here code to destroy and create agents (or to reset agents and space if agents are not recreated after each simulation). Lastly, the modeler needs to implement a method `eval` which evaluates quantitative observations on the current simulation state. Figure 2 (right) shows that the new methods are responsible for instructing the `Scheduler` and setting the `Space`.

Listing 1 shows a snippet of the famous Boids flocks model [63]. We consider it in Sect. 5. Integrating the model with MultiVeStA required only minor changes, including adding the required methods. Changes are localized in the class `BoidFlockers`, which extends the class `Model`. We can see that the invocation of method `make_agents` has been moved to `set_simulator_for_new_simulation`. Also, `eval` supports two observations (discussed in Sect. 5). The class for agents did not require any change, and therefore it is not shown.

5 Transient Analysis of the Boids Flocker Model

We now demonstrate the integration of MultiVeStA and Mesa using a case study from the Mesa model library [52]. In particular, we use the Boids flocker model [63] mentioned in Sect. 4, focusing on transient analysis.

5.1 Model Description

This is an ABM model where agents are birds,[4] while the emergent behavior regards the formation of flocks. The model was created by Craig Reynolds [63],

[3] Instructions and replicability material the Mesa models used in this paper is available at https://github.com/andrea-vandin/MultiVeStA/wiki/Integration-with-Mesa.

[4] Actually, unspecified flocking animals named *boids*, therefore the name of the model https://www.red3d.com/cwr/boids/.

```
class BoidFlockers(mesa.Model):
  def __init__(self, parameters):
    self.parameters = parameters
    #part of model initialization moved to
        set_simulator_for_new_simulation
    #self.make_agents()

  def make_agents(self):
    ...

  def step(self): #Method already implemented, no need to change
    self.schedule.step()

  #Code below is specific for integration with MultiVeStA

  def set_simulator_for_new_simulation(self,rnd_seed):
    random.seed(rnd_seed)
    self.random.seed(rnd_seed)
    self.reset_randomizer(rnd_seed)
    #part of model initialization moved here from the constructor
    self.make_agents()
    ...

  def eval(self, obs):
    if obs == 'avg_distance_from_centroid':
      ...
    elif obs == 'avg_visible_neighbors':
      ...
```

Listing 1. Code snippet of Mesa's Boids flockers integrated with MultiVeStA

and later encoded in several frameworks, including Mesa[5]. By encoding just a few simple rules for the behavior of birds, the model generates sophisticated and realistic flocks behaviors. The agents move in a 2-dimensional continuous space. That is, space is not abstracted in discrete structures (grids or similar), and the height at which birds fly is ignored. The model includes a number of parameters. We consider a 100×100 space with 100 birds that move at speed 2 (at each step, perform movements of 2 units of space). Birds have a vision radius of 10 (the radius within which each bird perceives neighboring birds), and separation 1 (birds within radius 1 are considered to be too close).

Each bird interacts with the others according to three simple *rules*: *Rule 1* (Cohesion), *Rule 2* (Separation) and *Rule 3* (Alignment) described in [57]. The execution of each rule gives a direction (i.e., a 2-dimensional vector) towards which performing the next move. The actual direction is computed by combining these three vectors via a weighted sum, where weights are three model parameters 0.03, 0.015, and 0.05, for Cohesion, Separation, and Alignment, respectively. The actual movement is computed by multiplying the resulting direction by the speed of the bird. *Rule 1* steers birds towards the *center of mass* (or centroid, i.e., the average position) of the perceived neighbors; *Rule 2* steers birds away from birds within the separation threshold; *Rule 3* forces birds to match the

[5] https://github.com/projectmesa/mesa-examples/tree/main/examples/boid_flockers.

(average) direction of the perceived neighbors. For all parameters, we use the values as given in the model.

A simulation step of the model, triggered by method `step` in Listing 1, consists in iterating all agents to apply the three rules and perform a movement. In each step, birds are iterated with a different randomly generated ordering. Figure 3 shows three snapshots of a simulation with these parameters as provided by the GUI of Mesa. Figure 3 (left) shows an initial step where birds are randomly placed in the space. Agents with less than two neighbors are red, while the others are green. Figure 3 (center) shows the state after 100 steps, containing three larger flocks (bottom, center, center-right). Figure 3 (right) shows that the three larger flocks are still present after 105 steps, and they have slightly moved.

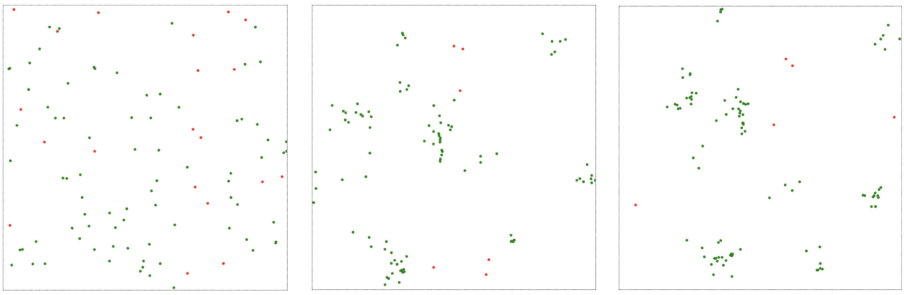


Fig. 3. Two snapshots from Mesa GUI for a simulation of the Boids model.

The Mesa repository contains several files inherent to this model, e.g., graphical components. We only need to use the core one containing the model specification (model.py). The changes to this file have been briefly discussed in Sect. 4.

5.2 State Observations and Predator

We use MultiVeStA to estimate properties on flock behaviors. In particular, in the `eval` function sketched in Listing 1 we encoded two observations to study aspects of the flocks. Observation `avg_distance_from_centroid` goes through each agent, computes the distance of the agent from the average of position of its neighbors, and then returns the average of this distance for all agents. This gives us an indication about how 'compact' is the *flock* perceived by each bird. Instead, observation `avg_visible_neighbors` computes the average number of neighbors perceived by each agent. A low value denotes a scenario where birds are scattered, possibly in several small flocks, while a high value denotes a scenario where most of the birds belong to few (possibly one) large flocks.

Inspired by [25], we also include the presence of *predators*. That is, at a given point in time, a predator appears causing the birds to disperse. As suggested in [57], when the *predator event* happens, we change the sign of the weight of the Cohesion vector (*Rule 1*). This should cause the flocks to scatter. After some

```
1  ObsAtStep(step,obs) =
2          if ( s.rval("steps") == step )
3                  then s.rval(obs)
4                  else next(ObsAtStep(step,obs)) fi ;
5  eval parametric(E[ ObsAtStep(x,"avg_distance_from_centroid")],
                     E[ MyProperty(x,"avg_visible_neighbors")],x,1,5,601)
       ;
```

Listing 2. MultiQuaTEx query for the Boids flocker model

time, the effect of the predator disappears, i.e., we reset the sign of the weight. We modified method `step` in Listing 1 such that the predator arrives at step 200, and its effect disappears at step 400. After executing these two steps, we iterate trough all birds flipping the sign of the considered weight.

5.3 Analysis Settings

We conducted an experiment using the described parameters. We use MultiVeStA to study the expected value of the two observations from Sect. 5.2 in every fifth step from 1 to 601 (1, 6, ...). We have set 600 as time horizon to have three periods of same length (200 steps) without and with the effect of the predator. We instructed MultiVeStA in doing so by using the MultiQuaTEx [66] query in 2. MultiQuaTEx [66] is a practitioner-oriented query language, or quantitative logic, to express quantitative temporal formulas. See [3] for a discussion on the relation among the transient fragments of MultiQuaTEx, CSL, and PCTL. Lines 1–4 specify a *recursive temporal operator* `ObsAtStep(step,obs)`, while line 5 specifies that the steps to consider are all from 1 to 601, with increment 5 (120 steps of interest overall). Instead, the two observations `obs` to be studied in these steps are `avg_distance_from_centroid` and `avg_visible_neighbors` discussed in Sect. 5.2. Line 5 instructs MultiVeStA to evaluate the expected value of several instances of the temporal operator. The temporal operator will be unrolled for each of the 120 steps of interest and for the two observations, (`ObsAtStep(1,obs)`, ..., `ObsAtStep(601,obs)`), leading to 240 independent estimations. Each instance will undergo the shown recursive if-then-else computation: whenever the simulation is in the corresponding step, `s.rval(obs)` will trigger the execution of method `eval` in Listing 1, otherwise the `next` in line 4 will trigger the execution of one step of simulation (method `next`), and the operator will be re-evaluated in the obtained simulation state.

We ask MultiVeStA to estimate each of these 240 expected values with 95% confidence intervals (CIs) of width at most 1. At the same time, we impose a maximum number of simulations (600). MultiVeStA will keep performing rounds of 30 simulations, each time evaluating the estimates and CIs. As soon as the CI of an estimate gets smaller than 1, it is not considered anymore. MultiVeStA completes its analysis as soon as all estimates reach the required CI width, or upon performing 600 simulations (possibly returning a CIs larger than 1). Clearly, each of the 240 estimations might require a different number of simulations, and in some cases 600 simulations might not be enough to bound the CI

width to 1. The choice of setting $\delta = 1$ and maximum number of simulations 600 has been made arbitrarily for the sake of presentation. In practice, this could be tuned as follow: one could impose a smaller δ, no limit on the number of simulations, and ask MultiVeStA to produce CSV files with estimations and CI widths computed after every round of 30 simulations. Then, one could either wait for all 240 estimations to reach the required CI width, or terminate the analysis when the latter round or simulations gave satisfactory CI widths.

We also conducted a second experiment where we increased the Coherence factor (the weight of *Rule 1*) from the default value of 0.03 to 0.09.

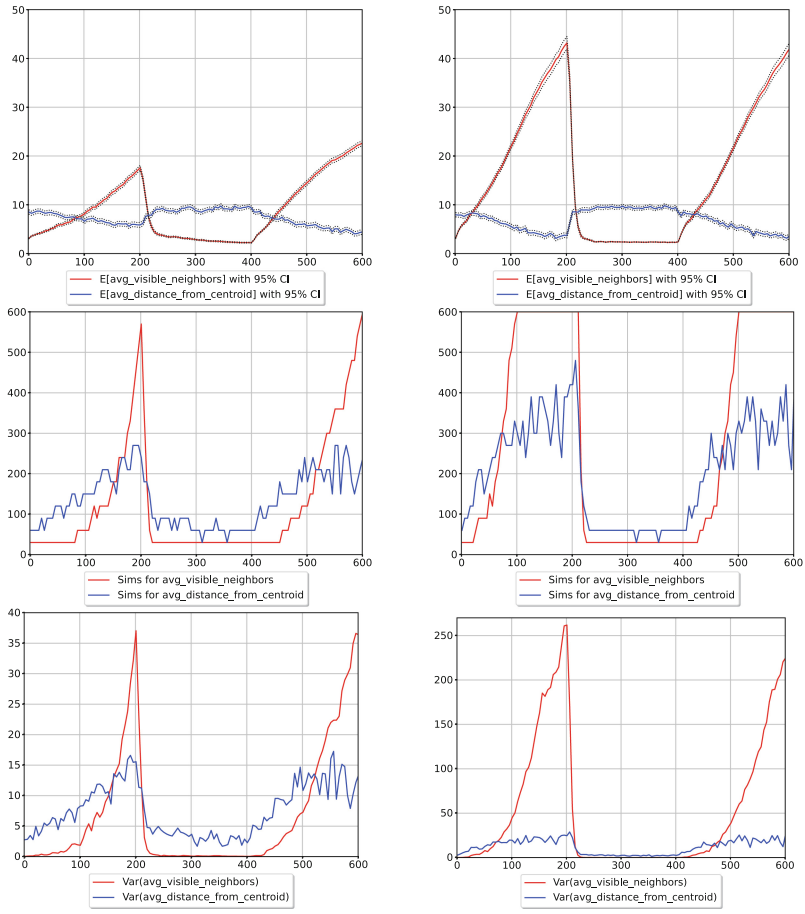


Fig. 4. MultiVeStA results for Listing 2. The predator arrives at step 200, and its effect ends at step 400. Left: results, number of simulations, and sample variance for the default cohesion weight 0.03. Right: same as for cohesion weight 0.09.

5.4 Results

The results of both experiments are shown in Fig. 4.

Default parameterization. We start discussing the results for default parameters in Fig. 4 (left). We start considering the red lines, connected to the neighbors counts. The top plot shows the estimated and corresponding confidence intervals computed for each of the 120 estimations. The middle plot shows that obtaining these CIs required a variable number of simulations: 600 at step 601, 570 at step 201 (right after the predator event). Otherwise, much fewer simulations were necessary, with a tendency of growing at the growing of the estimated value. From the bottom plot we can see that the sample variance follows a similar trend. Indeed, as discussed, the higher the variance, the higher has to be the number of simulations to bound the CI width.

We now move to the blue lines, connected to the distance from the center of the neighbors. The top plot shows that the estimates are more stable over time than for the other property. In particular, we note a decreasing trend in the time frames when there is no effect of the predator, and a more stable trend under the effect of the predator. The lower variability of this property is confirmed by the middle and bottom plots.

By combining the information from the two properties, we can give the following interpretation on the flock behavior: in the initial states of each simulation, where agents are randomly placed in the space, birds can perceive on average 4 neighbors, that is, birds might be scattered in several small flocks. While applying the three rules, the number of perceived neighbors grows to about 17 until the predator comes. At the same time, the average distance from the centers of the perceived neighbors decreases, even if slowly. This means that birds tend to group in fewer, larger, and more compact flocks. The arrival of the predator has a strong impact on the dynamics. The flip of sign of the coherence factor leads birds to isolate, going back to a number of perceived neighbors similar to the one of birds randomly distributed in the space. Also, birds tend to be further away from the (few) neighbors. As soon as the effect of the predator disappears, the flocking behavior resumes, going up to 25 neighbors on average.

Higher Coherence Factor. We discuss the results for coherence factor 0.09 in Fig. 4 (right). The top plot shows a much stronger flocking behaviors, leading to 45 neighbors on average on step 200, about 3 times those in Fig. 4 (left-top). Similarly, the impact of the predator is stronger. By looking at Fig. 4 (right-middle) and Fig. 4 (right-bottom), we confirm the higher variability of this property than in Fig. 4 (left). Actually, we can see that in many steps it has been necessary to run all admitted 600 simulations. Somehow surprisingly, the reparameterization does not seem to have an impact on the dynamics of property `avg_distance_from_centroid`. In fact, the blue lines in Fig. 4 (right-top) are similar to the ones in Fig. 4 (left-top).

Figure 5 zooms in of the widths of the obtained CIs. We can see that for the default parameterization we always satisfy the constraint on the CI width

within 600 simulations. Instead, this was not the case for `avg_visible_neighbors` in Fig. 4 (right), where 600 simulations lead to CIs of width up to 2.6.

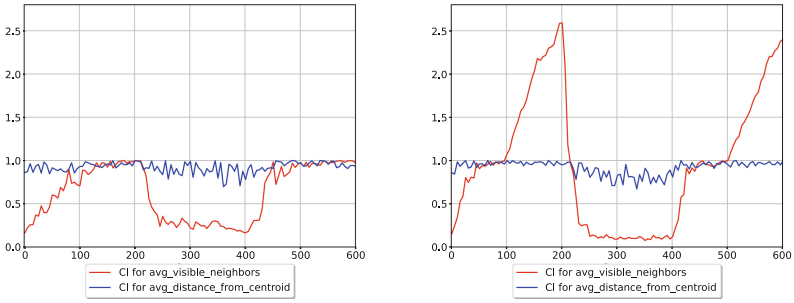


Fig. 5. CI widths for the estimations in Fig. 4. Left: results for the default cohesion weight 0.03. Right: same as left for cohesion weight 0.09.

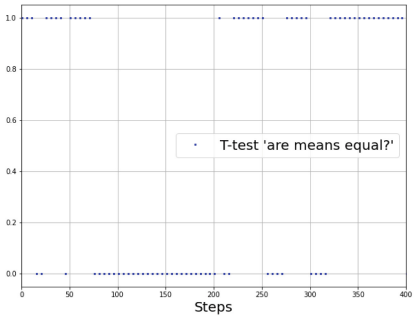


Fig. 6. Counterfactual analysis: T-test for each considered step for `avg_distance_from_centroid` from the analyses in Fig. 4 (top).

Counterfactual Analysis. We discussed how the change in parameters led to clear changes in the dynamics of `avg_visible_neighbors`. Instead, it is not clear if there has been a significant impact on `avg_distance_from_centroid`. We can formally answer this question by using the counterfactual analysis capabilities of MultiVeStA discussed in Sect. 2.3. For this property, we run a t-test for each considered step to check whether the results obtained for the two parameterizations are *significantly different from a statistical point of view*. The results are shown in Fig. 6. We can see that the higher coherence factor did not have an impact in about half of the steps, the 1-dots at the beginning and the end. Instead, the central steps (from about 75 to about 200) have been more considerably affected. Indeed, e.g., in such points the blue lines in Fig. 4 (top-right) seem to be more stable than the ones in Fig. 4 (top-left).

6 Steady-State Analysis of Schelling Segregation Model

We now consider another classic ABM model from the Mesa model library, the Schelling model [65], focusing on steady-state analysis.

6.1 Model Description and State Observations

The second model used to evaluate the integration of Mesa with MultiVeStA is the famous Schelling segregation model [65]. As for the Boids model, it has been encoded in several frameworks, including Mesa[6]. It considers two groups of people (agents), *red* and *blue*, representing two groups of *similar* individuals. The red agents are the *majority group*, while the blue ones are the *minority* one. The model is famous for its emergent behavior of segregation: it shows how a modest preference for residing close to neighbors of the same group can result in the minority group being segregated in a few areas [65].

Space is abstracted in a square matrix, with people living in cell locations (at most one agent per cell). Agents are *happy* if at least a certain number (by default 3) of the 8 neighboring positions contain people of their group. The dynamics of the model only regards unhappy agents that keep picking a random empty cell until happy (agents start moving again if changes in the neighborhood make them unhappy). If the system runs for long enough, the model stabilizes showing areas with communities of same group, de facto segregating the minority one. A number of parameters can be tweaked to modify the behavior of the agents:

- The width and height of the `Space` in which agents will interact;
- *Homophily*, i.e., the tendency of individuals to connect with others who share similar characteristics or attributes. It is the minimum number of neighbors of same color required by an agent to be happy;
- The density factor, dictating the number of agents in the system. It is the probability of adding an agent in each position during initialization;
- The minority percentage, representing the percentage of agents that belong to the minority group. When an agent is created, it is assigned to the minority group with this probability.

We use default values as available in the model: we consider a 20×20 matrix with density 0.8, minority percentage 0.2, and homophily 3.

Figure 7 gives a graphical representation of the model using Mesa's GUI. Figure 7 (left) shows an initial configuration of where agents are randomly scattered in the space. We note how about 20% of the cells are empty, as dictated by density, and how about 20% of the agents are blue, as dictated by minority percentage. Figure 7 (right) shows a state obtained after about 900 steps of simulation. We can already see partial effects of segregation: the minority group has gathered in 4 areas.

Similarly to the Boids case, the Mesa models repository contains several files inherent to this model, but we only had to focus on the core file containing the

[6] https://github.com/projectmesa/mesa-examples/tree/main/examples/schelling.

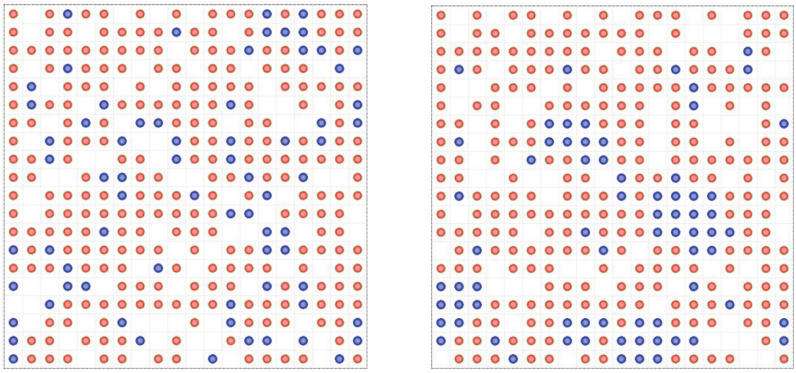


Fig. 7. Two snapshots from Mesa GUI for a simulation of the Schelling model.

model specification (model.py). In this file, we had to do only minor changes like those discussed for Boids in Sect. 4. We only had to modify the class `Schelling` which extends `Model` by moving part of the initialization code in the new method to reset the simulator, and by adding the method `eval`. In the latter, we encode a simple observation `ratioHappy` which computes the ratio of happy agents. To this evaluation, we also add a mild white noise (we add a value sampled from a standard normal distribution). This does not affect the estimated value (as we use mean 0), and has the benefit of making our steady-state algorithms more robust to cases of convergence to deterministic fixed points.

```
1  Obs(obs) = s.rval(obs);
2  eval autoRD(E[ Obs("ratioHappy")]);
3
4  Obs(obs) = s.rval(obs);
5  eval autoBM(E[ Obs("ratioHappy")]);
```

Listing 3. MultiQuaTEx queries for the Schelling model

6.2 Analysis Settings and Results

Our goal is to confirm that the model, and in particular the ratio of happy agents, stabilizes over time. We do this using the steady-state analysis capabilities of MultiVeStA presented in Sect. 2.2. We use the MultiQuaTEx queries in Listing 3, evaluated with a 95% CI with width at most 0.1. For the first query, MultiVeStA runs `autoRD` with default hyperparameters from [76] (e.g., for estimating the warmpup period we use 128 batches, with initial batch size of 8, meaning that the first check for termination of the warmup period is after 128*8=1024 steps). We find that the warmup period is terminated after the first check. MultiVeStA then starts performing rounds of 30 simulations, evaluating for each the average value of `ratioHappy` in the steps 1025–2048 (the horizontal means for time horizon 2048 discarding the warmup period). It turns out that 30 simulations are enough to get an estimate (0.98) with CI of accepted width.

By running `autoBM`, we obtain similar results: warmup is estimated to end after 1024 steps. Then, the single simulation continues to compute the estimate. The algorithm `autoBM` performs 1024 steps more on the simulation, getting to step 2048. Here, the CI was still too large (0.13). The algorithm then performed 2048 steps more, i.e., it doubled the time horizon getting to 4096 steps. We got an estimate of 1.02 with CI of accepted width.

We now check whether our methodology for ergodicity diagnostics (see Sect. 2.4) identifies violations on the assumptions required for steady-state analysis. Both algorithms terminate, and the two estimates are 0.04 away, which is less than the imposed CI width (0.1). Therefore, the first to checks do not signal any violation. The horizontal means computed by `autoRD` pass the normality test required in the second check of our methodology. In conclusion, our ergodicity diagnostics methodology does not signal violations, therefore we do not have reason to suspect that the performed steady-state analyses were unreliable.

7 Conclusions

Agent-based modeling (ABM) is a powerful modeling paradigm to obtain complex emergent behaviors from simple interactions of many heterogeneous agents. Analyzing ABMs requires designing complex simulation experiments. In computer science, statistical model checking (SMC) is a popular family of automated analysis techniques for simulation models. We presented an integration of Mesa, an ABM framework popular in the social sciences, with MultiVeStA, a statistical model checker that can be integrated with existing simulators without requiring the modelers to encode their models in third-party frameworks or languages. The integration enhanced the analysis capabilities of Mesa models with SMC, automating part of the analyses of interest making them more reliable. We demonstrated the integration using two seminal ABMs, the Boids flocks model and the Schelling segregation model.

We identify a number of possible future directions: MultiVeStA has been recently extended with process-oriented data science techniques, process mining, to explain graphically and intuitively the results computed with SMC [18]. We plan to extend these results to successfully apply them to Mesa models, helping Mesa modelers in debugging, fixing and refining their models. Another issue regards steady-state analysis. Often, models from the social sciences do not have just one equilibria, a steady-state, but possibly multiple ones. We will investigate the development of techniques to identify and estimate multiple equilibria of ABM models. It might also be interesting to relate the presented tool-chaining with existing ones involving SMC and other formalisms targeted to analyse emergent behavior (e.g. [49,53]). Finally, we will consider improving the integration mechanisms of MultiVeStA with simulators using advanced patterns for service-oriented applications [80].

Acknowledgments. Work partially supported by SMaRT COnSTRUCT (CUP J53C24001460006), in the context of FAIR (PE0000013, CUP B53C22003630006) under the National Recovery and Resilience Plan (Mission 4, Component 2, Line of Investment 1.3) funded by the European Union - NextGenerationEU. I thank Roberto Casaluce and Antonio Corallo for discussions on the integration of MultiVeStA and Mesa, and Roberto also for creating a preliminary version of Fig. 2. I thank Marco Pangallo and Daniele Giachini for discussions on the Schelling model and on economic ABMs in general.

References

1. van der Aalst, W.M.: Process Mining, 2nd edn. Springer (2016)
2. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Trans. Model. Comp. Simul. **28**(1), 6:1–6:39 (2018)
3. Agha, G.A., Meseguer, J., Sen, K.: PMaude: rewrite-based specification language for probabilistic object systems. In: Cerone, A., Wiklicky, H. (eds.) QAPL 2005. ENTCS, vol. 153, no. 2, pp. 213–239. Elsevier (2006)
4. Alexopoulos, C., Goldsman, D.: To batch or not to batch? ACM Trans. Model. Comput. Simulat. **14**(1), 76–114 (2004)
5. Alexopoulos, C., Seila, A.F.: Implementing the batch means method in simulation experiments. In: Proceedings of WSC 1996, pp. 214–221 (1996)
6. An, G., Wilensky, U.: From artificial life to in silico medicine. In: Komosinski, M., Adamatzky, A. (eds.) Artificial Life Models in Software, pp. 183–214. Springer, London (2009)
7. Bartoletti, M., Chiang, J.H., Junttila, T., Lluch-Lafuente, A., Mirelli, M., Vandin, A.: Formal analysis of lending pools in decentralized finance. In: Proceedings of ISoLA 2022. LNCS, vol. 13703, pp. 335–355. Springer (2022)
8. ter Beek, M.H., Legay, A., Lafuente, A.L., Vandin, A.: Quantitative security risk modeling and analysis with RisQFLan. Comput. Secur. **109**, 102381 (2021)
9. ter Beek, M.H., Legay, A., Lluch-Lafuente, A., Vandin, A.: A framework for quantitative modeling and analysis of highly (re)configurable systems. IEEE Trans. Software Eng. **46**(3), 321–345 (2020)
10. Belzner, L., De Nicola, R., Vandin, A., Wirsing, M.: Reasoning (on) service component ensembles in rewriting logic. In: Iida, S., Meseguer, J., Ogata, K. (eds.) Specification, Algebra, and Software. LNCS, vol. 8373, pp. 188–211. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54624-2_10
11. Bengtsson, J., Larsen, K.G., Larsson, F., Pettersson, P., Yi, W.: UPPAAL - a tool suite for automatic verification of real-time systems. In: Alur, R., Henzinger, T.A., Sontag, E.D. (eds.) Hybrid Systems III. LNCS, vol. 1066, pp. 232–243. Springer, Heidelberg (1995). https://doi.org/10.1007/BFB0020949
12. Bortolussi, L., Milios, D., Sanguinetti, G.: Machine learning methods in statistical model checking and system design - tutorial. In: RV 2015. LNCS, vol. 9333, pp. 323–341. Springer, Cham (2015)
13. Bortolussi, L., et al.: CARMA: collective adaptive resource-sharing Markovian agents. In: QAPL. EPTCS, vol. 194, pp. 16–31 (2015)
14. Bottazzi, G., Giachini, D.: Far from the madding crowd: collective wisdom in prediction markets. Quantit. Finance **19**(9), 1461–1471 (2019)
15. Brown, D.G., Page, S., Riolo, R., Zellner, M., Rand, W.: Path dependence and the validation of agent-based spatial models of land use. Int. J. Geogr. Inf. Sci. **19**(2), 153–174 (2005)

16. Bruni, R., Corradini, A., Gadducci, F., Lluch-Lafuente, A., Vandin, A.: Modelling and analyzing adaptive self-assembly strategies with Maude. Sci. Comput. Program. **99**, 75–94 (2015)
17. Caiani, A., Godin, A., Caverzasi, E., Gallegati, M., Kinsella, S., Stiglitz, J.E.: Agent based-stock flow consistent macroeconomics: towards a benchmark model. J. Econ. Dyn. Control **69**, 375–408 (2016)
18. Casaluce, R., Burattin, A., Chiaromonte, F., Lafuente, A.L., Vandin, A.: White-box validation of quantitative product lines by statistical model checking and process mining. JSS **210**, 111983 (2024). https://doi.org/10.1016/j.jss.2024.111983
19. Casaluce, R., Burattin, A., Chiaromonte, F., Vandin, A.: Process mining meets statistical model checking: towards a novel approach to model validation and enhancement. In: Cabanillas, C., Garmann-Johnsen, N.F., Koschmider, A. (eds.) BPM Workshops, pp. 243–256. Springer, Cham (2023)
20. Casaluce, R., Burratin, A., Chiaromonte, F., Lluch-Lafuente, A., Vandin, A.: Enhancing threat model validation: a white-box approach based on statistical model checking and process mining. In: Breve, B., Desolda, G., Deufemia, V., Spano, L.D. (eds.) Proceedings of the First International Workshop on Detection and Mitigation of Cyber Attacks that Exploit Human vuLnerabilitiES (DAMOCLES 2024) Co-located with 17th International Conference on Advanced Visual Interfaces (AVI 2024), Arenzano (Genoa), Arenzano, 4 June 2024. CEUR Workshop Proceedings, vol. 3713, pp. 9–20. CEUR-WS.org (2024). https://ceur-ws.org/Vol-3713/paper_2.pdf
21. Ciancia, V., Latella, D., Massink, M., Paškauskas, R., Vandin, A.: A tool-chain for statistical spatio-temporal model checking of bike sharing systems. In: ISOLA 2017 (2017)
22. Conway, R.W.: Some tactical problems in digital simulation. Manage. Sci. **10**(1), 47–61 (1963). https://doi.org/10.1287/mnsc.10.1.47
23. Corradini, F., Fornari, F., Polini, A., Re, B., Tiezzi, F., Vandin, A.: A formal approach for the analysis of BPMN collaboration models. JSS **180**, 111007 (2021)
24. David, A., Larsen, K.G., Legay, A., Mikucionis, M., Poulsen, D.B.: Uppaal SMC tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (2015). https://doi.org/10.1007/S10009-014-0361-Y
25. De Nicola, R., Di Stefano, L., Inverso, O., Valiani, S.: Modelling flocks of birds and colonies of ants from the bottom up. JSTTT **25**(5), 675–691 (2023). https://doi.org/10.1007/s10009-023-00731-0
26. Dehnert, C., Junges, S., Katoen, J., Volk, M.: A storm is coming: a modern probabilistic model checker. In: Majumdar, R., Kuncak, V. (eds.) CAV 2017, Part II. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31
27. Delli Gatti, D., Grazzini, J.: Rising to the challenge: Bayesian estimation and forecasting techniques for macroeconomic agent based models. J. Econ. Behav. Organiz. **178**, 875–902 (2020)
28. Dosi, G., Fagiolo, G., Roventini, A.: Schumpeter meeting Keynes: a policy-friendly model of endogenous growth and business cycles. JEDC **34**(9), 1748–1767 (2010). https://doi.org/10.1016/j.jedc.2010.06.018
29. Dosi, G., Roventini, A.: More is different... and complex! the case for agent-based macroeconomics. J. Evolution. Econ. **29**(1), 1–37 (2019)
30. Effken, J.A., Carley, K.M., Lee, J.S., Brewer, B.B., Verran, J.A.: Simulating nursing unit performance with orgahead: strengths and challenges. Comput. Inf. Nurs. **30**(11), 620 (2012)

31. Fagiolo, G., Roventini, A.: Macroeconomic policy in DSGE and agent-based models. Revue de l'OFCE **124**, 67–116 (2012)
32. Fagiolo, G., Giachini, D., Roventini, A.: Innovation, finance, and economic growth: an agent-based approach. J. Econ. Interac. Coord. **15**(3), 703–736 (2019). https://doi.org/10.1007/s11403-019-00258-1
33. Fagiolo, G., Roventini, A.: Macroeconomic policy in DSGE and agent-based models redux: New developments and challenges ahead. JASSS **20**(1) (2017)
34. Galpin, V., Georgoulas, A., Loreti, M., Vandin, A.: Statistical analysis of CARMA models: an advanced tutorial. In: Johansson, B., Jain, S. (eds.) Proceedings of WSC. pp. 395–409. IEEE (2018). https://doi.org/10.1109/WSC.2018.8632456
35. Gilmore, S., Reijsbergen, D., Vandin, A.: Transient and steady-state statistical analysis for discrete event simulators. In: IFM, pp. 145–160. Springer, Cham (2017)
36. Gilmore, S., Tribastone, M., Vandin, A.: An analysis pathway for the quantitative evaluation of public transport systems. In: IFM (2014)
37. Grimm, V., Railsback, S.F.: Individual-Based Modeling and Ecology. Princeton University Press (2013)
38. Herd, B., Miles, S., McBurney, P., Luck, M.: Quantitative analysis of multiagent systems through statistical model checking. In: Engineering Multi-agent Systems, pp. 109–130. Springer, Cham (2015)
39. Herd, B., Miles, S., McBurney, P., Luck, M.: A Monte Carlo model checker for multiagent-based simulations. In: Multi-agent Based Simulation XVI, pp. 37–54. Springer, Cham (2016)
40. Hussain, F., Langmead, C.J., Mi, Q., Dutta-Moscato, J., Vodovotz, Y., Jha, S.K.: Automated parameter estimation for biological models using Bayesian statistical model checking. BMC Bioinformatics **16**(17), S8 (2015)
41. Kazil, J., Masad, D., Crooks, A.: Utilizing python for agent-based modeling: the mesa framework. In: Social, Cultural, and Behavioral Modeling, pp. 308–317. Springer, Cham (2020)
42. Kelton, W.D., Law, A.M.: An analytical evaluation of alternative strategies in steady-state simulation. Oper. Res. **32**(1), 169–184 (1984). https://doi.org/10.1287/opre.32.1.169
43. Kroiß, C.: Simulation and statistical model checking of logic-based multi-agent system models. In: Agent and Multi-agent Systems: Technologies and Applications, pp. 151–160. Springer, Cham (2014)
44. Kwiatkowska, M., Norman, G., Parker, D.: Prism: probabilistic symbolic model checker. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) Computer Performance Evaluation: Modelling Techniques and Tools, pp. 200–204. Springer, Heidelberg (2002)
45. Law, A.M., Carson, J.S.: A sequential procedure for determining the length of a steady-state simulation. Oper. Res. **27**(5), 1011–1025 (1979). https://doi.org/10.1287/opre.27.5.1011
46. Law, A.M., Kelton, D.M.: Simulation Modeling and Analysis, 5th edn. McGraw-Hill Higher Education (2015). http://www.averill-law.com/simulation-book/
47. Legay, A., Lukina, A., Traonouez, L.M., Yang, J., Smolka, S.A., Grosu, R.: Statistical model checking. In: Computing and Software Science: State of the Art and Perspectives, pp. 478–504. Springer, Cham (2019)
48. Legay, A., Sedwards, S., Traonouez, L.: Rare events for statistical model checking an overview. In: Proceedings of RP 2016. LNCS, vol. 9899, pp. 23–35. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-45994-3_2

49. Legay, A., Traonouez, L.: Statistical model checking with change detection. LNCS Trans. Found. Master. Chang. **1**, 157–179 (2016). https://doi.org/10.1007/978-3-319-46508-1_9
50. Macy, M.W., Willer, R.: From factors to actors: computational sociology and agent-based modeling. In: Annual Review of Sociology, pp. 143–166 (2002)
51. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. **18**(1), 50–60 (1947). https://doi.org/10.1214/aoms/1177730491
52. Mesa: Mesa Models Repository. https://github.com/projectmesa/mesa-examples. Accessed 15 May 2024
53. Mignogna, A., Mangeruca, L., Boyer, B., Legay, A., Arnold, A.: SOS contract verification using statistical model checking. In: Larsen, K.G., Legay, A., Nyman, U. (eds.) Proceedings 1st Workshop on Advances in Systems of Systems, AiSoS 2013, Rome, 16 March 2013. EPTCS, vol. 133, pp. 67–83 (2013). https://doi.org/10.4204/EPTCS.133.7
54. Nicola, R.D., Ferrari, G., Pugliese, R.: Locality based linda: Programming with explicit localities. In: Bidoit, M., Dauchet, M. (eds.) TAPSOFT 1997. LNCS, vol. 1214, pp. 712–726. Springer, Cham (1997). https://doi.org/10.1007/BFB0030636
55. Nicola, R.D., Stefano, L.D., Inverso, O.: Multi-agent systems with virtual stigmergy. SCP **187**, 102345 (2020). https://doi.org/10.1016/J.SCICO.2019.102345
56. Pangallo, M., et al.: The unequal effects of the health-economy trade-off during the covid-19 pandemic. Nat. Hum. Behav. **8**(2), 264–275 (2024). https://doi.org/10.1038/s41562-023-01747-x
57. Parker, C.: Pseudocode of Boids Flocks Model. https://vergenet.net/~conrad/boids/pseudocode.html. Accessed 15 May 2024
58. Pianini, D., Sebastio, S., Vandin, A.: Distributed statistical analysis of complex systems modeled through a chemical metaphor. In: HPCS, pp. 416–423 (2014)
59. Poledna, S., Miess, M.G., Hommes, C.H.: Economic forecasting with an agent-based model (2020). Available at SSRN 3484768
60. Rasmussen, J.I., Behrmann, G., Larsen, K.G.: Complexity in simplicity: flexible agent-based state space exploration. In: TACAS 2007, pp. 231–245. Springer (2007)
61. Rasmussen, J.I., Behrmann, G., Larsen, K.G.: Complexity in simplicity: flexible agent-based state space exploration. In: TACAS 2007. LNCS, vol. 4424, pp. 231–245. Springer, Cham (2007). https://doi.org/10.1007/978-3-540-71209-1_19
62. Reinhardt, O., Warnke, T., Uhrmacher, A.M.: A language for agent-based discrete-event modeling and simulation of linked lives. ACM TOMACS **32**(1), 6:1–6:26 (2022). https://doi.org/10.1145/3486634
63. Reynolds, C.W.: Flocks, herds and schools: a distributed behavioral model. In: Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, pp. 25–34 (1987)
64. Richiardi, M.G., Richardson, R.E.: Jasmine: a new platform for microsimulation and agent-based modelling. Int. J. Microsimulat. **10**(1), 106–134 (2017)
65. Schelling, T.C.: Dynamic models of segregation †. J. Math. Sociol. **1**(2), 143–186 (1971). https://doi.org/10.1080/0022250X.1971.9989794
66. Sebastio, S., Vandin, A.: MultiVeStA: statistical model checking for discrete event simulators. In: ValueTools 2013, pp. 310–315. ICST/ACM (2013)
67. Secchi, D., Seri, R.: Controlling for *false negatives* in agent-based models: a review of power analysis in organizational research. Comput. Math. Organ. Theory **23**(1), 94–121 (2016). https://doi.org/10.1007/s10588-016-9218-0

68. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 202–215. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_16
69. Steiger, N.M., Lada, E.K., Wilson, J.R., Joines, J.A., Alexopoulos, C., Goldsman, D.: Asap3: a batch means procedure for steady-state simulation analysis. ACM TOMACS **15**(1), 39–73 (2005)
70. Tafazzoli, A., Wilson, J.R., Lada, E.K., Steiger, N.M.: Performance of skart: a skewness-and autoregression-adjusted batch means procedure for simulation analysis. INFORMS J. Comput. **23**(2), 297–314 (2011)
71. Tesfatsion, L., Judd, K.L.: Handbook of Computational Economics: Agent-Based Computational Economics. Elsevier (2006)
72. Uhrmacher, A.M., Weyns, D. (eds.): Multi-agent systems - simulation and applications. In: Computational Analysis, Synthesis, and Design of Dynamic Systems. CRC Press/Taylor & Francis (2009). https://doi.org/10.1201/9781420070248
73. Valente, M.: Laboratory for simulation development: LSD. Tech. rep., LEM Working Paper Series (2008)
74. Vandin, A., ter Beek, M.H., Legay, A., Lluch-Lafuente, A.: QFLan: a tool for the quantitative analysis of highly reconfigurable systems. In: FM (2018)
75. Vandin, A., Giachini, D., Lamperti, F., Chiaromonte, F.: Multivesta: statistical analysis of economic agent-based models by statistical model checking. In: From Data to Models and Back - 10th International Symposium, DataMod 2021, Revised Selected Papers. LNCS, vol. 13268, pp. 3–6. Springer (2021). https://doi.org/10.1007/978-3-031-16011-0_1
76. Vandin, A., Giachini, D., Lamperti, F., Chiaromonte, F.: Automated and distributed statistical analysis of economic agent-based models. JEDC **143**, 104458 (2022). https://doi.org/10.1016/j.jedc.2022.104458
77. Welch, B.L.: The generalization ofstudent's' problem when several different population variances are involved. Biometrika **34**(1/2), 28–35 (1947)
78. Whitt, W.: The efficiency of one long run versus independent replications in steady-state simulation. Manage. Sci. **37**(6), 645–666 (1991)
79. Wilensky, U.: NetLogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston (1999). http://ccl.northwestern.edu/netlogo/
80. Wirsing, M., et al.: Sensoriapatterns: augmenting service engineering with formal analysis, transformation and dynamicity. In: Proceedings of ISoLA 2008. Communications in Computer and Information Science, vol. 17, pp. 170–190. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88479-8_13
81. Younes, H.L.: Probabilistic verification for "black-box" systems. In: CAV 2015, pp. 253–265. Springer, Heidelberg (2005)