Proceedings of the 2020 Winter Simulation Conference
K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, and R. Thiesing, eds.

PRAGMATIC LOGIC-BASED SPATIO-TEMPORAL PATTERN CHECKING IN PARTICLE-BASED MODELS

Andreas Ruscheinski
Anja Wolpers
Philipp Henning
Tom Warnke
Fiete Haack
Adelinde M. Uhrmacher

Institute for Visual and Analytic Computing
University of Rostock
Albert-Einstein-Str. 22
Rostock, 18059, GERMANY

ABSTRACT

Particle-based simulation is a powerful approach for modeling systems and processes of entities interacting in continuous space. One way to validate a particle-based simulation is to check for the occurrence of spatio-temporal patterns formed by the particles, for example by statistical model checking. Whereas spatio-temporal logics for describing spatio-temporal patterns exist, they are defined on discrete rather than continuous space. We propose an approach to bridge this gap by automatically translating the output of continuous-space particle-based simulations into an input for discrete-space spatio-temporal logics. The translation is parameterized with information about relevant regions and their development in time. We demonstrate the utility of our approach with a case study in which we successfully apply statistical model-checking to a particle-based cell-biological model. A Java implementation of our approach is available under an open-source license.

1 INTRODUCTION

Particle-based simulation is a powerful approach for modeling systems and processes of entities interacting in continuous space (Andrews 2009).

It has been successfully applied in various application domains, for example, cell biology (Schöneberg et al. 2014; Bitting and Uhrmacher 2017), physics (Werner et al. 2011), and crowd simulation (Guy et al. 2011). Such simulations are driven by the interactions of particles among themselves and with the environment. The interesting aspect of the output, however, is often spatio-temporal patterns formed by a group of particles. As particle-based simulations typically produce lots of output data, defining such patterns and checking for their occurrence is not trivial.

The ability to make and automatically check statements about the simulation output, however, is a valuable tool for the validation of simulations. Statistical model-checking, for example, is a validation method that uses the output of individual simulation runs. It applies hypothesis testing to a number of simulation runs to infer statistically reliable statements about the simulation (Agha and Palmskog 2018). Statistical model-checking is a well-established tool in many application areas, focusing mostly on simulation behavior in time. Different temporal logics and corresponding checking algorithms have been proposed (Fages and Rizk 2007; Maler and Nickovic 2004). Several approaches have extended this to spatio-temporal behavior by augmenting temporal logics with spatial operators. However, these approaches are defined on discrete space and, therefore, not directly applicable to particle-based simulations.

In this paper, we propose a conceptual framework that bridges the gap between particle-based simulation and spatio-temporal logics. Our core contribution is a method to translate the output from a particle-based simulation to a discretized representation that is usable for logics defined on discrete space. The translation is parameterized with information about relevant geometric elements and their development in time. We demonstrate the utility of our approach with an implementation that is based on the Signal Spatio-Temporal Logic (SSTL) (Bortolussi and Nenzi 2014) and that allows executing statistical model-checking experiments with particle-based simulation.

2 PARTICLE-BASED SIMULATION AND SPATIO-TEMPORAL PATTERNS

In a particle-based simulation, the simulation model typically consists of a description of the particles in terms of their initial location, their movement, and how the particles interact with each other. Depending, on the application and the simulators, e.g., ML-Force (Köster et al. 2019) or NetLogo (Tisue and Wilensky 2004), the particles can also be equipped with custom attributes to represent characteristics such as the size of or the mass of the particles or even more abstract features like the damage of a particle (Dalmasso et al. 2017). These attributes can then be used to specify particle movement that depend on the particle attributes, e.g., the movement of a differently charged particles in an electric field, or interaction rules that modify the attributes, e.g., colliding particles are merged to a particle with combined mass and size. During the simulation, the particle locations are updated based on how they move. Therefore, the movement might calculated from forces acting on particles or from the "personality" of a particle (Guy et al. 2011).

Further, the collision of the particles has to be detected and resolved according to the interaction or reaction rules, respectively. The concrete particle attributes, the forces, and the interaction rules but also the number of particles or whether the particles are located in 2D- or 3D-space in the simulation strongly depend on the actual application of the particle-based simulation.

The possibility to interpret different kinds of entities as particles yields to a variety of applications for particle-based simulations. For example in the field of quantum chemistry, particle-based simulations can be used to calculate properties of chemical reactions (Werner et al. 2011). Therefore, we represent atoms as particles. Consequently, their attributes correspond to atomic properties, e.g., the atomic mass or the atomic orbital. The forces that determine the particle movement can then be calculated from the laws of quantum mechanics. A quite different example of particle-based simulations can be found in the field of crowd simulation, where we want to simulate the behavior of crowds formed by individuals. Guy et al. (2011) use particles to represent persons. Each particle is equipped with a set of attributes like "preferred speed" or "planning horizon" to represent the personality of a person. The movement of the particle is then calculated from the goal of the person, their personality, and the particles around them.

However, the output of these simulations typically comprises the particle attributes, i.e., the particle locations and the custom attributes, for each observation step in the simulation . For example, in a simulation with three particles in 2D (see Figure 1) observed at $t = 1$ and $t = 10$, the output consists of the particle locations (x and y) at the observation step (see Table 1) . The interesting aspects of output are often the patterns formed by the particles in the simulation.

Next, we describe a selection of four typical spatio-temporal patterns that can be formed by particles. Later, we revisit these patterns to discuss how our pragmatic pattern checking approach can be used to check whether these patterns occurred in the simulation.

Particle Growing and Shrinkage In this pattern, the particles grow or shrink in their size during the simulation (see Figure 2). Also, the particles might change their growing/shrinking behavior during the simulation in terms of their growing/shrinking rate. Consequently, this pattern requires that the particles are equipped with an attribute representing their size, e.g., the radius or the diameter. This pattern is typically produced by particles emitting or absorbing other particles or processes "inside" particles.

An example from the field of biology is the vesicle transport model (Heinrich and Rapoport 2005) and its particle-based adaption (Köster et al. 2019). In this model organelles in a cell grow and shrink due to the emission and absorption of vesicles. A more detailed description of the model can be found in

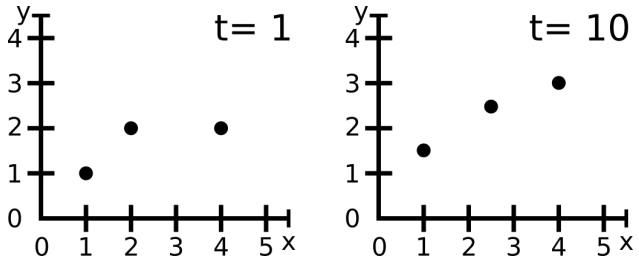

Figure 1: Simulation of three particles observerd at $t = 1$ and $t = 10$

Table 1: Output of the simulation of the three particles with their 2D locations at $t = 1$ and $t = 10$

t	x	y
1	1	1
1	2	2
1	4	2
10	1	1.5
10	2.5	2.5
10	4	3

the case-study in Section 5. Another occurrence of this pattern is e.g. the growth of a cell over the time followed by an abrupt shrinking if the cell divides (Milde et al. 2014).

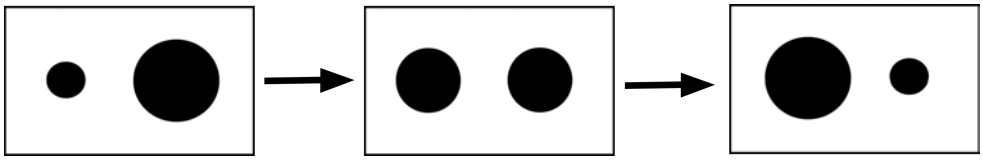

Figure 2: Schema of the particle growing and shrinking pattern

Clustering of Particles A cluster is a group of particles (see Figure 3). To specify what constitutes a cluster, i.e., the clustering criterion, has been identified as a difficult problem (Estivill-Castro 2002). The reason for this is that "clustering is in the eye of beholder" (Estivill-Castro 2002) which also yields a variety of algorithms to automatically detect clusters. For example, a cluster might be specified based on the required density of particles (Kriegel et al. 2011), i.e., the number of particles in a region has to be higher than the number of particles in the neighborhood of the region. Thus, the modeler has to take the knowledge about the particle behavior and to the system simulation into account for specifying the clustering criterion. Since this pattern describes a specific arrangement of particles in space it does not require any additional attributes of the particles. However, a more fine-grained criterion can be specified by taking additional attributes into account, e.g., a cluster formed of particles of a certain size.

One case where the clustering pattern can be observed is at the formation of actin filaments in cells growing on a structured surface e.g. an implant. This was shown in a particle-based model (Bittig et al. 2014) of actin filaments accumulate at certain areas of a structured surface due to the affinity of anchor proteins (integrin) to these regions. Also, the higher concentration of integrin influences the activity of proteins which shortens the actin filaments as seen in the wet-lab data. Another example of this pattern can be found in a particle-based version of the predator-prey model where the reproduction of the prey yields to the formation of a cluster followed by the accumulation by predators (Mohapatra and Mahapatra 2019).

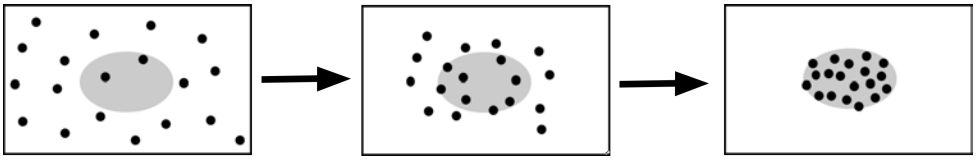

Figure 3: Schema of the clustering of particles pattern

Particle Oscillation In general, we perceive particle oscillation as a repetitive movement of particles between two or more regions during the simulation (see Figure 4). Therefore, we can distinguish between

undamped and damped oscillation. In the undamped oscillation, the repetitive movement occurs between fixed regions whereas in the damped oscillation the regions move closer to each other (Landau et al. 2013). In the simulation, this pattern typically occurs whenever the forces in the simulation model attract and repeal the particles in time-intervals to certain regions in the simulation. Since this pattern describes a characteristic movement of the particles it does require any additional attributes of the particles.

The pattern of oscillating particles can be found in the formation of MinD protein rings on the membrane of E. coli bacteria. The PDE model by Huang et. al (Huang et al. 2003) was the first one that was able to reproduce this oscillation of proteins. To take stochastic effects into account the original model was converted into a particle-based approach (Kerr et al. 2005) where the oscillation can be observed, too. Interestingly, the oscillation of the proteins is not explicitly noted in the model, but instead emerges from the set of reaction rules. A different example of this pattern is the movement of magnetic particles in an alternating magnetic field. In contrast to the oscillation of MinD proteins, the pattern is explicitly noted in the model as it typically includes the magnetic field as the driving force of the oscillation (Zhou et al. 2019).

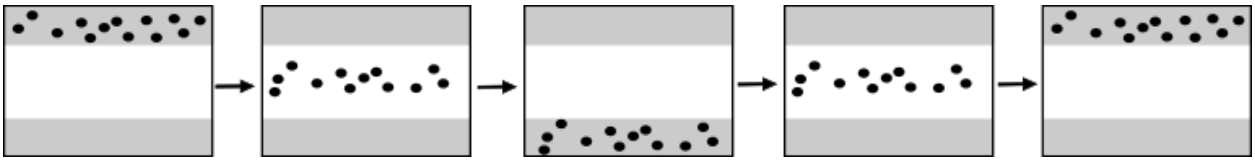

Figure 4: Schema of the particle oscillation pattern

Particle sources and Particle sinks A source is a region in the simulation where new particles are created whereas a sink is a region where particles get destroyed (see Figure 5). However, for the sources we require a particle attribute allowing us to distinguish between "old" and "new" particles, e.g., a timestamp at which the particle was created. Using this attribute, we can then check whether new particles are only created at source regions. For sinks we require a particle attribute allowing us to distinguish between "living" and "dead" particles, e.g., a boolean flag. Using this attribute, we can then check whether dead particles only occur in sink regions.

The example of a source/sink can be found in the same model vesicle transport model as presented at the growing/shrinking pattern and the case study (see Section 5) were organelles emit or absorb particles. This pattern can also be found in the emission of pheromones by cells or a radioactive material which radiates α or β particles.

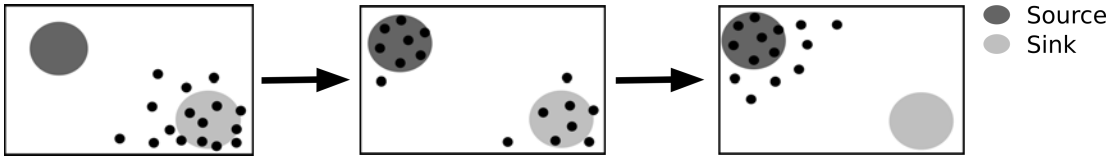

Figure 5: Schema of the sources and sinks pattern

3 PRAGMATIC CONTINUOUS-SPACE PATTERN MONITORING

Spatio-temporal patterns like the ones described in the previous section can be an important ingredient for model validation. For example, a model might only be deemed valid if it reproduces specific behavior known from other models or real-world experiments (Leye et al. 2009). Therefore, the modeler might visualize the output data and observe the particles to confirm whether the pattern has occurred (face validation). However, there are several reasons against manually assessing the output data. First, human assessment is necessarily subjective, which might compromise resulting statements about the model validity. In particular, reproducing the result would require repeating the (subjective) assessment exactly. Second, in complex simulation experiments, the number of runs to assess might become very large. Third, the pattern might not

be easily recognizable in the output (for example, because the pattern is formed by few of overall many particles). Thus, it is important to provide modelers with an objective and automated method to check the model output against some pattern specifications.

One example of applying such a method is statistical model-checking, also called runtime verification, assertion checking, or monitoring (Donzé et al. 2013). Here several simulation runs are checked against a property specification to infer the probability that a random simulation run satisfies the specification via hypothesis testing.

For example, we can use *temporal logics*, such as Signal Temporal Logic (STL) or Linear Temporal Logic (LTL), to specify the property and use corresponding checking algorithms to test whether a run satisfies the property (Maler and Nickovic 2004). However, temporal logics typically operate on scalar output variables, for example, to compare an output to a constant or two outputs to each other. This allows statements such as "Output b will be greater than output a during the first 100 time units", formalized in STL as $\boldsymbol{G}_{[0,100]}(a > b)$, where $\boldsymbol{G}$ is the bounded *always* operator. To additionally allow statements about spatial properties, temporal logics have been extended to *spatio-temporal logics*.

One example of a spatial extension of a temporal logic is the Signal Spatio-Temporal Logic (SSTL), which is based on STL (Bortolussi and Nenzi 2014). It operates on a weighted graph of locations, and the truth of propositions (like $a > b$ above) depends not only on the time point of evaluation but also on the location. Accordingly, SSTL adds operators like the bounded *everywhere* operator $\boxdot$ to express spatial properties. For example, the formula $\boxdot_{[0,5]}(a > b)$ states that output a shall be greater than output b in all locations that are reachable in the location graph with a path cost of at most 5 from a given starting location.

To apply spatio-temporal logics to particle-based simulation, we bridge the gap between the continuous space in which the particles move and the discrete space on which the logic formulas are defined.

Therefore, our approach allows us to automatically transform a specification of a spatio-temporal pattern and the particle information collected from the simulation into a representation specific to a selected spatio-temporal logic, based on which the pattern can then be checked. Therefore, our approach relies on a three-step process in which we transform the specification of a spatio-temporal pattern and the particle data into a normalized representation based on which features for the pattern are calculated and target representation is generated (see Figure 6).

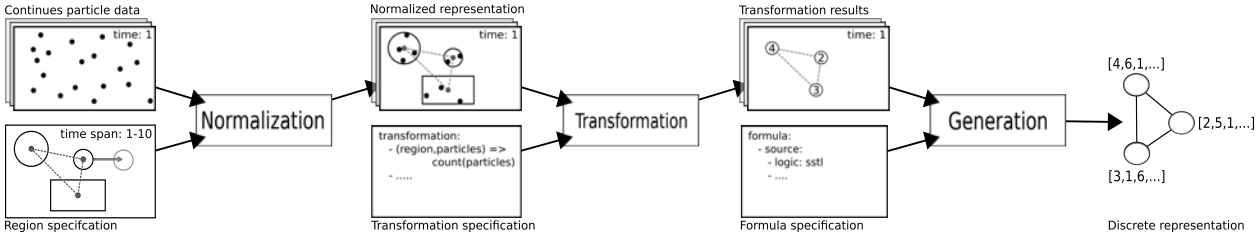

Figure 6: Overview of the three-step process of our method

In the following, we describe our approach for 2D patterns. However, we shortly discuss the required adaptions to also support 3D in the conclusion of the paper.

3.1 Specifying spatio-temporal patterns

In addition to a spatio-temporal formula as described in the previous section, our approach requires two more inputs. First, a description of *regions*, which are dedicated areas in 2D space that exist over time and are connected with each other. Second, a set of *transformations* that calculate *features* for each region and observation time based on the particles contained in the region. Then spatio-temporal patterns can be expressed as spatio-temporal formulas over these features.

Regions Each region is specified by its area, a reference point, a period where the region exists, and the links to other regions. In general, we allow the area to be an arbitrary shape, e.g., a circle, a rectangle, or

polygons, as long it possible to check whether a particle lies inside the area by using the particle locations. Additionally, the shape is parameterized by a position in 2D space and its size. However, some patterns require moving regions, e.g., to describe a moving cluster or a damped oscillation. For this, we adopted a keyframe-based method to specify different positions and sizes of a shape within the period. Based on the difference between the keyframes the concrete position and size can be interpolated for each timestamp within the period. Finally, the regions need to have a reference point inside the area based on which the links to other regions are defined. A graphical overview of a region specification is shown in Figure 7.

Later, we will use the regions to represent the locations in a discrete representation for the spatio-temporal logics. The links and the distances between the regions are then be used to link the locations.

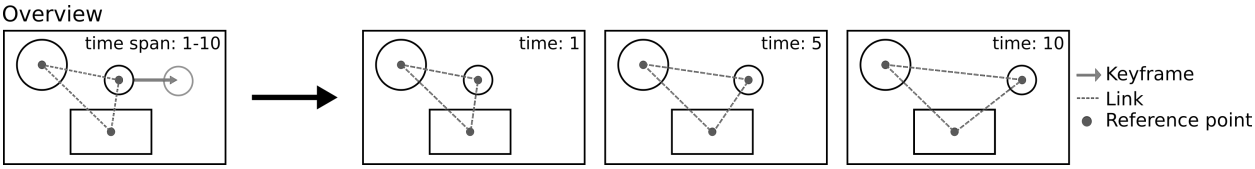

Figure 7: Graphical overview of a region specification

Transformations The transformations are functions that calculate the features based on the region and their particles inside. This information can then be used to calculate features like, e.g., the density of the particles or the number of particles. Also, the confined particles can be filtered to calculate features for patterns that are formed by a certain class of particles, e.g., all particles with a size larger than a given threshold. Finally, we also allow to calculate "abstract" features, i.e., features that are calculated independently from the region or the particles given as arguments. For example, we can use this mechanism to calculate the difference between the number of particles of two explicitly given regions.

3.2 Normalization, Transformation and Generation

Given the specification as described in the previous section and the particle data, we generate a discrete representation in three steps. In the first step, we create a normalized representation, i.e., a mapping, which allows getting the confined particles for each region for each observation time. In the second step, we calculate the features by applying the transformations to all regions in the normalized representation. In the third step, we then generate the discrete representation for the spatio-temporal logic by using the features.

For the normalization step, we first have to collect the existing regions at each observation time from the particle data and then calculate the particles inside these regions. Therefore, we iterate for each observation step in the particle data over the specified regions and check whether the observation step lies within the period of the region. Whenever a region is specified by keyframes, we also interpolate the concrete position and size at each observation step. Thereafter, we calculate the particles inside the collected regions.

In the transformation step, we apply the transformation to regions collected from the previous step. By applying the transformations, the normalized representation becomes a mapping allowing to get the features for each region at each observation time.

Based on this mapping we can then generate a discrete representation for the spatio-temporal logic the generation step of our approach. The discrete representation might differ between different spatio-temporal logics due to different underlying semantics.

For example, in SSTL space is represented as a weighted graph whereas the output at the location and time are represented as a spatio-temporal trace (Bortolussi and Nenzi 2014). Thus, in our approach, the specified regions become the nodes and the links between the regions become the edges in the weighted graph. These edges are then weighted by the distance between the regions, i.e., the distance between their reference points. Then transformation results calculated on a region become the spatio-temporal trace of the corresponding location.

4 IMPLEMENTATION DETAILS

We implemented a prototype of our approach using the Java programming language (the software is available under https://git.informatik.uni-rostock.de/mosi/GREASE/smock). Therefore, we followed the pipeline architecture pattern (Vermeulen et al. 1995). According to the pattern, we structured the prototype into individual processing elements while interfaces make the inputs and outputs of the elements explicit. This clear separation of concerns between the processing elements also allows integrating new processing elements into the pipeline without changing other elements. We used the elements to represent the steps of our approach and also included dedicated elements for parsing the particle data and the pattern specification and checking the spatio-temporal formula.

The particle data is parsed into a lookup table, allowing to access the particles and their attributes. Therefore, the table records are storing the observation times of the particles, their locations, and the custom attributes. In our current implementation, we provide CSV parser for the particle data.

The pattern parser uses the pattern specification to create internal representations of the regions, transformations, and spatio-temporal formulas. The implementation currently only supports rectangles, quads, and circles as shapes. However, we also allow specifying a shape subtracting shapes from each other. We adopted YAML (Ben-Kiki, Oren and Evans, Clark and döt Net, Ingy 2001) as the specification language for the spatio-temporal patterns (see Listing 1). By default the reference point is at the center of a region and the regions are fully connected. However, the modeler can override the reference point and links in the specification.

To interpolate the keyframes in the normalization, we calculate a transformation matrix from the different sizes and positions of the regions at its keyframes. This matrix is then applied to the region by using linear interpolation for observation steps between two keyframes.

In our current implementation, we provide a generator and model checker for SSTL.

5 EVALUATION

In the following, we demonstrate our method using the prototype to check the sources and sinks and the particle growing and shrinking patterns in a vesicular transport model. By this, we also show how our method can be integrated into a statistical model-checking algorithm to make a statistically reliable statement about the particle behavior.

5.1 Case Study: A Vesicular Transport Model

The vesicle transport model (Heinrich and Rapoport 2005) describes how organelles in a cell can communicate by vesicles without losing their characteristic protein composition. The original ODE model consists of 2 compartments (representing the organelles) that emit vesicles with different coatings at a rate proportional to the compartment size. These coatings affect the movement of the vesicles and their ability to fuse with the compartments. Due to the differences in the behavior of the vesicles, the compartments can communicate and still maintaining their protein composition.

To get a deeper understanding of the properties of the vesicle transport, we translated the deterministic ODE model into a stochastic particle-based model in ML-Force (Köster et al. 2019). In the simulation, the particles are equipped with attributes representing their location in 2D, velocity, mass, radius, and the force acting on the particle. The radius of the particles varies over time for the compartments whereas the radius of the vesicles is always 25 nm. To simulate the directed movement of the vesicles between the compartments, an external force is added to the vesicles that determine their movement in addition to the random movement by diffusion. The emission of vesicles from the compartments and the fusion of the vesicles into the compartments in the case of a collision is defined in a set of reaction rules. In Figure 8, we show a schema of the compartments and vesicles at the beginning and in a later state of the simulation.

Due to the behavior of the compartments and the vesicles, the model should show two patterns. First, the compartments should grow and shrink if vesicles are incorporated or emitted, and thus the initial radius

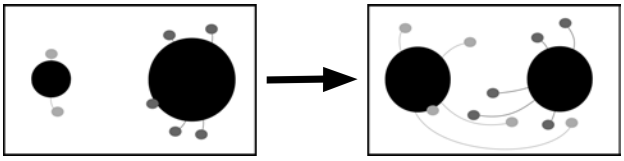

Figure 8: Overview of the vesicular transport model. The compartments are shown as black circles whereas the vesicles are shown as small gray circles. The lines to the vesicles are indicating movement of the particles.

difference between them should decrease over time. Second, the compartments should act as sources and sinks for the vesicles. In the simulation, we simulated 274500 steps corresponding to simulation time of ≈ 0.02 s and observed the locations and the size of particles every 500 steps.

Pattern specification The declarative specification of the pattern is shown in Listing 1.

Listing 1: Declarative specification of the patterns

```
regions:
  - name: compartment1
    period: [0,20000000]
    shape: Circle
    keyframes:
      - begin: 0
        properties:
          center: [1349.5,2249.5]
          radius: 400
      - begin: 20000000
        properties:
          center: [1349.5,2249.5]
          radius: 500
  - name: compartment2
    period: [0,20000000]
    shape: Circle
    keyframes:
      - begin: 0
        properties:
          center: [3849.5,2249.5]
          radius: 600
      - begin: 20000000
        properties:
          center: [3849.5,2249.5]
          radius: 500
  - name: rest
    ...
transformations:
  - name: newParticles
    formula: (region , particles) -> particles.stream().filter(record->record.getAttribute("new").equals("true")).count()
  - name: lastParticles
    formula: (region , particles) -> particles.stream().filter(record->record.getAttribute("last").equals("true")).count()
  - name: compartmentDifference
    ...
formulas:
  - name: growing-and-shrinking
    logic: sstl
    targets: [compartment1]
    specification: (compartmentDifference < 250) U[0,20000000](
      (compartmentDifference < 170) U[0,20000000](
        (compartmentDifference < 100) U[0,20000000] (compartmentDifference < 10)))
  - name: source
    logic: sstl
    targets: [compartment1, compartment2]
    specification: G[0,20000000] ((newParticles > 0) -> ([[]][0,1300] (newParticles == 0)))
  - name: sink
    ...
```

In the pattern specification, we used three different regions. Two of those are enclosing the compartments, while the other represents the whole area in the simulation without the compartments. The compartment regions are represented as circles fixed at the location of the compartments from the simulation. To specify the radius of the compartment regions, we also have to take the granularity of the particle data into account. For example, a new particle might be created between two observation steps thus appears to be created outside of the components. Thus, we increased our radius for the compartments' regions and added 100 nm

from the original radius from the simulation model. Further, we use keyframes to specify the different sizes of the compartments regions at the beginning and the end of the simulation.

To calculate the features for the patterns, we use transformations to count the number of new particles, count the particles removed from the simulation between two observation steps, and calculate the radius difference between the two compartments. For this, we calculated the radius difference as a "abstract" feature just directly referring to the compartments in the corresponding regions. Thus, we filtered for particles with a radius > 25 nm to remove the vesicles and calculated the radius difference of the remaining compartment particles.

We specified three spatio-temporal formulas to check the sources, the sinks, and the growing and shrinking pattern individually. To check sources, we specified a formula which states "whenever the number of new particles is greater than 0, the number of particles in the surrounding regions reachable with a path cost of at most 1300 should be 0" and checked the formula for the two compartment regions. By setting maximal path cost to 1300 only the region without the compartments is reachable from the compartment regions. Similarly, the sinks are specified. For the growing and shrinking behavior, we specified a formula that states that the compartment difference should decrease over time and evaluated the formula on a single compartment region.

Checking the patterns with statistical model checking Due to the underlying stochastics of the simulation model, we integrated the prototype of our pattern checking method into a statistical model checking algorithm implemented in Python (a software artifact running the case study available under https://doi.org/10.5281/zenodo.3904203). Therefore, we followed the method described in Ciancia et al. (2016). The outline of the statistical model checking with the included pattern checking method is shown in pseudo-code in Algorithm 1.

Algorithm 1: Outline of the statistical model-checking algorithm in pseudocode

```
Data: A simulation and a pattern specification
Result: Has the pattern occured?
evidence = [];
while additional-evidence-required(evidence) do
    data = run-simulation();
    pattern-results = check-patterns(data, specification);
    evidence.add(pattern-results);
return analyse-evidence(evidence)
```

The statistical model checking algorithm executes simulation runs until enough evidence is collected from the pattern evaluation to make a statistically reliable statement about the patterns by using the Sequential Probability Ratio Test (Agha and Palmskog 2018). We ran the simulations using the ML-Force simulator with the simulation model and collect the particle data for each run. Thereafter, we processed the data into a single CSV file containing the observation times, particle locations, the size of the particle, and calculated additional attributes for new particles created and the particles removed between two observations. The patterns are then checked using the pattern specification and the CSV file. The checking results were then added to the evidence. This process was repeated until enough evidence was collected. Finally, the overall result was calculated from the overall evidence.

We ran the Python script on an Intel Core i7-7700K CPU @ 4.20GHz with 8 cores and collected the evidence from 7 simulation runs in parallel. The algorithm executed and analyzed 28 simulation runs to collect enough evidence that the patterns are formed in the simulation. In total our algorithm took ≈ 3000 s and generated ≈ 550 MB of simulation data. Of the total run time 80% were spent on the simulation whereas 20% were spent on the data processing and the pattern checking.

5.2 Specifying the other patterns

Next, we will give a short intuition about how the clustering and the oscillation pattern can be checked.

Clustering To check the clustering pattern, the modeler has to specify two regions. One region r_1 corresponds to the area where cluster forms while the other region r_2 represents the area surrounding r_1. In

this setting, a spatio-temporal formula then states a property about features that are calculated according to the clustering criterion. For example for a density-based clustering criterion, the transformation can calculate the density of the particles by dividing the number of particles within the regions by the area of the regions. The spatio-temporal formula than states a property like "the particle density in r_1 needs to be higher than the particle density in r_2".

Oscillation The oscillation of particles can be described by specifying the regions between which the particles oscillate and using a transformation to count the particles within the regions. The spatio-temporal formula must then express that the count of the particles oscillates between the regions. As discussed by Andreychenko et al. (2014), however, linear time temporal logics like the ones used in statistical model-checking can not express permanently repeating oscillation. One way to address this is to generate formulas that express a given number of oscillation cycles. See Fages and Rizk (2007) for an example.

6 RELATED WORK

Approaches for describing and finding spatiotemporal patterns with logics have been developed in several domains, e.g., for movement of mobile agents (Jing-De Cheng 2008) and mobile processes in graphs (Nicola et al. 2007). In addition to formulations based on logics, spatiotemporal patterns have been expressed with database queries (Sakr and Güting 2014; Warnke and Uhrmacher 2016). However, also the particle movement can be analyzed and used to make a statement about the particle behavior (Laube et al. 2005). In contrast to our work, these approaches focus on the interaction of individual objects rather than patterns formed by objects in the simulation.

Nenzi et al. (2015) state that in many cases the continuous space can abstracted by grids or meshes to create a discrete space in which spatio-temporal model checking can be applied. In our approach this abstraction is made explicit by the regions in continuous space based on which a discrete representation is created. Moreover, we provide extra flexibility for specifying the regions by allowing arbitrary shapes and provide keyframes to move and scale the regions over time.

7 CONCLUSION

In this paper, we described a method to check whether 2D spatio-temporal patterns are formed in particle-based simulation. For this, our method bridges the gap between particles moving and forming patterns in continuous space and spatio-temporal logics allowing to formally check spatio-temporal patterns in discrete space. Therefore, the particle data and a declarative specification of the spatio-temporal pattern is converted into a discrete representation by the normalization, transformation, and generation steps of our method. In the specifications, the regions of interest, the features based on which the pattern can be detected, and the spatio-temporal formula checking whether the pattern has occurred are made explicit. By this, the specification can be reused for other particle-based models and also enables to reproduce the checking results. In the case study, we integrated our approach into a statistical model checking framework to make a statistically reliable statement about the patterns for the validation of the simulation model. However, our method can also be used to find regions where the patterns occur, e.g., we can search for clusters in the simulation by evaluating a corresponding spatio-temporal formula in different regions.

To extend our approach to 3D patterns, we have to include the specification 3D shapes, such as spheres, cubes, and cuboids, as regions. Consequently, this also requires to adopt a suitable method to check whether particles lies inside the volume. Further, this work can be combined with methods using pattern recognition (Townsend and Gong 2018) to automatically find regions in the simulation which then can be checked using our method.

ACKNOWLEDGMENTS

This work was funded by the research grants DFG UH 66/18 'GrEASE' and DFG CRC 1270 'Elaine'.

REFERENCES

Agha, G., and K. Palmskog. 2018. "A Survey of Statistical Model Checking". *ACM Transactions on Modeling and Computer Simulation* 28(1):6:1–6:39.

Andrews, S. S. 2009. "Accurate Particle-based Simulation of Adsorption, Desorption and Partial Transmission". *Physical Biology* 6(4):046015.

Andreychenko, A., T. Krüger, and D. Spieler. 2014. "Analyzing Oscillatory Behavior with Formal Methods". In *Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems*, 1–25. Springer Berlin Heidelberg.

Ben-Kiki, Oren and Evans, Clark and döt Net, Ingy 2001. "YAML Ain't Markup Language (YAML™) Version 1.2".

Bortolussi, L., and L. Nenzi. 2014. "Specifying and Monitoring Properties of Stochastic Spatio-Temporal Systems in Signal Temporal Logic". In *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*, VALUETOOLS '14, 66–73. Brussels, BEL: Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering.

Ciancia, V., D. Latella, M. Massink, R. Paškauskas, and A. Vandin. 2016. "A Tool-Chain for Statistical Spatio-Temporal Model Checking of Bike Sharing Systems". In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques*, 657–673. Springer International Publishing.

Dalmasso, G., P. A. M. Zapata, N. R. Brady, and A. Hamacher-Brady. 2017. "Agent-Based Modeling of Mitochondria Links Sub-Cellular Dynamics to Cellular Homeostasis and Heterogeneity". *PLOS ONE* 12(1):e0168198.

Donzé, A., T. Ferrère, and O. Maler. 2013. "Efficient Robust Monitoring for STL". In *Computer Aided Verification*, edited by N. Sharygina and H. Veith, 264–279. Berlin, Heidelberg: Springer Berlin Heidelberg.

Estivill-Castro, V. 2002. "Why so Many Clustering Algorithms". *ACM SIGKDD Explorations Newsletter* 4(1):65–75.

Fages, F., and A. Rizk. 2007. "On the Analysis of Numerical Data Time Series in Temporal Logic". In *Computational Methods in Systems Biology*, edited by M. Calder and S. Gilmore, 48–63. Berlin, Heidelberg: Springer Berlin Heidelberg.

Guy, S. J., S. Kim, M. C. Lin, and D. Manocha. 2011. "Simulating Heterogeneous Crowd Behaviors Using Personality Trait Theory". In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation - SCA '11*: ACM Press.

Heinrich, R., and T. A. Rapoport. 2005. "Generation of Nonidentical Compartments in Vesicular Transport Systems". *Journal of Cell Biology* 168(2):271–280.

Huang, K. C., Y. Meir, and N. S. Wingreen. 2003. "Dynamic Structures in Escherichia Coli: Spontaneous Formation of MinE Rings and MinD Polar Zones". *Proceedings of the National Academy of Sciences* 100(22):12724–12728.

Jing-De Cheng 2008. "Qualitative Spatio-temporal Reasoning About Movement of Mobile Agents/Objects". In *2008 International Conference on Machine Learning and Cybernetics*, Volume 6, 3341–3346.

Kerr, R. A., H. Levine, T. J. Sejnowski, and W.-J. Rappel. 2005. "Division Accuracy in a Stochastic Model of Min Oscillations in Escherichia Coli". *Proceedings of the National Academy of Sciences* 103(2):347–352.

Köster, T., P. Henning, and A. M. Uhrmacher. 2019. "Potential Based, Spatial Simulation of Dynamically Nested Particles". *BMC Bioinformatics* 20(607).

Kriegel, H.-P., P. Kröger, J. Sander, and A. Zimek. 2011. "Density-based Clustering". *WIREs Data Mining and Knowledge Discovery* 1(3):231–240.

Landau, L. D., A. I. Akhiezer, and E. M. Lifshitz. 2013. *General Physics: Mechanics and Molecular Physics*. Elsevier.

Laube, P., S. Imfeld, and R. Weibel. 2005. "Discovering Relative Motion Patterns in Groups of Moving Point Objects". *International Journal of Geographical Information Science* 19(6):639–668.

Leye, S., J. Himmelspach, and A. M. Uhrmacher. 2009. "A Discussion on Experimental Model Validation". In *11th International Conference on Computer Modelling and Simulation*: Institute of Electrical and Electronics Engineers IEEE.

Maler, O., and D. Nickovic. 2004. "Monitoring Temporal Properties of Continuous Signals". In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, edited by Y. Lakhnech and S. Yovine, 152–166. Berlin, Heidelberg: Springer Berlin Heidelberg.

Milde, F., G. Tauriello, H. Haberkern, and P. Koumoutsakos. 2014. "SEM++: A Particle Model of Cellular Growth, Signaling and Migration". *Computational Particle Mechanics* 1(2):211–227.

Mohapatra, S., and P. S. Mahapatra. 2019. "Confined System Analysis of a Predator-Prey Minimalistic Model". *Scientific Reports* 9(1).

Nenzi, L., L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink. 2015. "Qualitative and Quantitative Monitoring of Spatio-Temporal Properties". In *Runtime Verification*, 21–37: Springer International Publishing.

Nicola, R. D., J.-P. Katoen, D. Latella, M. Loreti, and M. Massink. 2007. "Model Checking Mobile Stochastic Logic". *Theoretical Computer Science* 382(1):42–70.

Bitting, A. T., and A. M. Uhrmacher. 2017. "ML-Space: Hybrid Spatial Gillespie and Particle Simulation of Multi-Level Rule-Based Models in Cell Biology". *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 14(6):1339–1349.

Bittig, A. T., C. Matschegewski, J. B. Nebe, S. Stählke, and A. M. Uhrmacher. 2014. "Membrane Related Dynamics and the Formation of Actin in Cells Growing on Micro-topographies: A Spatial Computational Model". *BMC Systems Biology* 8(1).

Sakr, M. A., and R. H. Güting. 2014. "Group Spatiotemporal Pattern Queries". *GeoInformatica* 18(4):699–746.

Schöneberg, J., A. Ullrich, and F. Noé. 2014. "Simulation Tools for Particle-based Reaction-diffusion Dynamics in Continuous Space". *BMC Biophysics* 7(1).

Tisue, S., and U. Wilensky. 2004. "NetLogo: A Simple Environment for Modeling Complexity". In *in International Conference on Complex Systems*, 16–21.

Townsend, R. G., and P. Gong. 2018. "Detection and Analysis of Spatiotemporal Patterns in Brain Activity". *PLOS Computational Biology* 14(12):1–29.

Vermeulen, A., G. Beged-Dov, and P. Thompson. 1995. "The Pipeline Design Pattern". In *Proceedings of OOPSLA'95 Workshop on Design Patterns for Concurrent, Parallel, and Distributed Object-Oriented Systems*. Citeseer.

Warnke, T., and A. M. Uhrmacher. 2016. "Spatiotemporal Pattern Matching in RoboCup". In *Multiagent System Technologies*, 89–104. Cham: Springer International Publishing.

Werner, H.-J., P. J. Knowles, G. Knizia, F. R. Manby, and M. Schütz. 2011. "Molpro: A General-purpose Quantum Chemistry Program Package". *Wiley Interdisciplinary Reviews: Computational Molecular Science* 2(2):242–253.

Zhou, J., S. Zhang, F. Tian, and C. Shao. 2019. "Simulation of Oscillation of Magnetic Particles in 3D Microchannel Flow Subjected to Alternating Gradient Magnetic Field". *Journal of Magnetism and Magnetic Materials* 473:32–41.

AUTHOR BIOGRAPHIES

ANDREAS RUSCHEINSKI is a Ph.D. candidate in the Modeling and Simulation group at the University of Rostock. His e-mail address is andreas.ruscheinski@uni-rostock.de.

ANJA WOLPERS is a bachelor student of Computer Science and a student assistant in the Modeling and Simulation group at the University of Rostock. Her e-mail address is anja.wolpers@uni-rostock.de.

PHILLIP HENNING is a Ph.D. candidate in the Modeling and Simulation group at the University of Rostock. His e-mail address is phillip.henning@uni-rostock.de.

TOM WARNKE is a Ph.D. candidate in the Modeling and Simulation group at the University of Rostock. His e-mail address is tom.warnke@uni-rostock.de.

FIETE HAACK is a postdoctoral researcher in the Modeling and Simulation group at the University of Rostock. His e-mail address is fiete.haack@uni-rostock.de.

ADELINDE M. UHRMACHER is a professor at the Institute of Visual and Analytic Computing, University of Rostock, and head of the Modeling and Simulation group. Her e-mail address is adelinde.uhrmacher@uni-rostock.de.