QUANTICOL

A Quantitative Approach to Management and Design of Collective and Adaptive Behaviours

quantiCOL

http://www.quanticol.eu

IR3.1

Scalable verification for spatial stochastic logics

Revision: 0.0; 18 Mar, 2016

Author(s): Luca Bortolussi (CNR), Vincenzo Ciancia (CNR), Stephen Gilmore (UEDIN), Jane Hillston (UEDIN), Diego Latella (CNR), Michele Loreti (IMT), Mieke Massink (CNR), Laura Nenzi (IMT), Rytis Paškauskas (CNR), Mirco Tribastone (IMT), Max Tschaikowski (IMT)

Due date of deliverable: Month 36 (March 2016)

Actual submission date: Mar 31, 2016

Nature: R. Dissemination level: PU

Funding Scheme: Small or medium scale focused research project (STREP)
Topic: ICT-2011 9.10: FET-Proactive 'Fundamentals of Collective Adaptive Systems' (FOCAS)
Project number: 600708

Coordinator: Jane Hillston (UEDIN)
e-mail: Jane.Hillston@ed.ac.uk
Fax: +44 131 651 1426

Part. no.	Participant organisation name	Acronym	Country
1 (Coord.)	University of Edinburgh	UEDIN	UK
2	Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologie della Informazione "A. Faedo"	CNR	Italy
3	Ludwig-Maximilians-Universität München	LMU	Germany
4	Ecole Polytechnique Fédérale de Lausanne	EPFL	Switzerland
5	IMT Lucca	IMT	Italy
6	University of Southampton	SOTON	UK
7	Institut National de Recherche en Informatique et en Automatique	INRIA	France

COOPERATION

Executive summary

This Internal Report describes the status of the work performed in the project on the extension of the theoretical foundations of scalable model-checking approaches with suitable notions of spatial verification. Various forms of scalable model-checking were developed during the first phase of Task 3.1 of WP3, including those based on mean-field and fluid flow techniques, and were presented in Deliverable 3.1. The focus of the present report is on forms of *spatial verification* based on *model-checking techniques* in which the effect of inhomogeneous spatial distribution of objects is taken into account.

Several spatial logics have been developed and explored. The Spatial Logic for Closure Spaces (SLCS) is based on the formal framework of *closure spaces*. The latter include topological spaces but also discrete spaces such as graphs and therefore form a promising candidate for a uniform framework to develop spatial logics that can be applied to the various spatial representations presented in the deliverables of WP2. Closure spaces come with a well-developed theory and some powerful operators that turned out to be very useful both for the definition of the semantics of SLCS and for the development of spatial and spatio-temporal model-checking algorithms. The spatial operators of SLCS have been also added to the Signal Temporal Logic (STL) to obtain Spatial Signal Temporal Logic (SSTL). In this case the spatial operators have been extended with spatial bounds based on distance measures and a quantitative semantics has been provided that is used to evaluate the *robustness* of the spatio-temporal formulas for signals.

Spatial and spatio-temporal performance analysis measures have been explored for a simple PALOMA model of a group of robots and for a bike sharing model based on Markov Renewal Processes. The latter provides a simulation model of bike-sharing systems that includes users as agents. It also generates simulation traces with spatio-temporal information on bike stations that can be analysed using the prototype spatio-temporal model checkers that have been developed in the context of the project.

A partial-differential approximation has been developed for spatial stochastic process algebra. The approach is validated on the well-known Lotka-Volterra model and shows an advantage in terms of computational efficiency compared to traditional numerical solutions.

Finally, for what concerns scalable verification, the innovative model-checking approaches based on fluid approximation and on-the-fly mean-field techniques have been finalised. For the latter a prototype model-checker, FlyFast, has been made available, which was described in Deliverable 5.2.

Contents

1 Introduction

Spatial aspects of computation are becoming increasingly relevant in Computer Science, especially in the field of *collective adaptive systems* and when dealing with systems distributed in physical space. Traditional formal verification techniques are well suited to analyse the temporal evolution of system models; however, properties of space are typically not taken into account explicitly. The global behaviour of CAS critically depends on interactions which are often local in nature. The aspect of locality immediately raises issues of spatial distribution of objects. Abstraction from spatial distribution may sometimes provide insights into the system behaviour, but this is not always the case. For example, consider a bike sharing system having several parking stations, and featuring twice as many parking slots as there are bikes in the system. Ignoring the spatial dimension, on average, the probability to find completely full or empty parking stations at an arbitrary location is very low; however, this kind of analysis may be misleading, as in practice some stations are much more popular than others, often depending on nearby points of interest. This leads to quite different probabilities to find stations completely full or empty, depending on the examined location.

In Deliverable 2.1 an extensive study has been presented on the mathematical modelling techniques that have been used to represent space, in particular when analysing CAS. These structures comprise several forms of discrete and continuous spatial models. In this Internal Report we focus on *reasoning* about spatial models, and in particular in doing so using *spatial logics*. The idea is that the right combination of spatial models and spatial logics may lead to the development of powerful formal verification techniques such as *spatial model-checking*, in analogy to the successful temporal model checking approaches developed in the past. This idea has been explored in several directions within the project. In one direction, spatial logics have been studied starting from well-known spatial logics for topological spaces [7]. In order to deal with both continuous and discrete spatial structures *closure spaces* [22] have been studied which are a generalisation of topological spaces. This has led to a *spatial logic for closure spaces* that has operators inspired both by operators known from closure spaces and by operators from temporal logics, but redesigned for use in spatial reasoning. Moreover, efficient spatial model-checking algorithms have been developed and implemented in prototype tools. Their combination with temporal logics has led to *spatio-temporal logics* and model checking, which has been applied on several case studies of the project. Section 2 and Section 3 give an overview of these developments.

In another direction, the spatial operators developed in the context of closure spaces have been added to a temporal logic for continuous signals (STL) [20, 29], leading to the *spatial signal temporal logic* (SSTL). The operators have also been enriched with spatial bounds based on distance measures and a quantitative semantics. The latter is used to evaluate the *robustness* of spatio-temporal formulas for signals. Work in this direction is described in Section 4.

Spatial verification raises also issues on what would be suitable and interesting performance measures to analyse in spatially distributed CAS. One could, for example, be interested in knowing the probabilities to find a particular agent in the various spatial locations in a steady state situation, leading to a probability distribution over space. Models of such systems often also include a mobility model for a large number of agents. Such models are notoriously difficult to analyse due to problems of scalability. A novel approach to address this issue is to use a spatial variant of continuous approximation. Both issues and results are briefly addressed in Section 5. Section 6 provides a brief update on the status of the work on scalable model checking techniques.

This Internal Report provides an intermediate update on the results and ongoing work performed as part of Task 3.1. Section 7 concludes the report by providing a work plan for Task 3.1 concerning scalable spatial verification for the last remaining year of the project. Further results will be reported in the forthcoming Deliverable 3.3 at the end of the project.

2 Spatial Model Checking Based on Closure Spaces

The mathematical structure of choice for spatial logics are very often *topological spaces*, possibly enriched with metrics, or other spatial features (see [7]). The use of abstract structures has the advantage to separate logical operators, such as neighbourhood, from the specific nature of space (e.g., the number of dimensions, or

$$
\begin{array}{rcll}
\Phi & ::= & a & [\text{ATOMIC PROPOSITION}] \\
 & | & \top & [\text{TRUE}] \\
 & | & \neg\Phi & [\text{NOT}] \\
 & | & \Phi \wedge \Phi & [\text{AND}] \\
 & | & \mathscr{N}\Phi & [\text{NEAR}] \\
 & | & \Phi \mathscr{S} \Phi & [\text{SURROUNDED}]
\end{array}
$$

Figure 1: SLCS syntax

the presence or absence of metric features, etc.). However, using topological spaces, it may be difficult to deal with discrete structures, such as finite graphs. In [22], *closure spaces*, which generalise topological spaces, are proposed as a unifying approach treating both topological spaces and graphs in a satisfactory way.

In [16] the authors proposed the logic SLCS (Spatial Logic for Closure Spaces), extending the topological semantics of modal logics to *closure spaces*. The work follows up on the research line of Galton [22] and Smyth and Webster [35], enhancing it with a modal logic perspective. Closure spaces (also called *Čech closure spaces* or *preclosure spaces* in the literature) are based on a single operator on sets of points, namely the *closure* operator, and are a generalisation of standard topological spaces. In addition, finite spaces and graphs are subclasses of closure spaces — technically, they belong to the class of *quasi discrete closure spaces* — and the graph-theoretical notion of neighbourhood coincides with the notion of neighbourhood defined in the context of closure spaces. Thus, closure spaces provide a uniform framework for the treatment of all major models of space, including those identified as relevant to CAS in the deliverables of WP2. This framework provides a set of useful basic abstract spatial operators (closure, interior, boundary and others) that provide a structured way to define higher level spatial logical operators. Moreover, they are suitable for the development of efficient spatial and (branching time) spatio-temporal model-checking algorithms in which these same closure space based operators play a role as well. Furthermore, metrics and distance functions can be added in an orthogonal way providing further spatial richness.

A *closure space* is composed of a set (of points) and a (closure) operator on subsets (of points), as specified by the following definition:

Definition 2.1 *A* closure space *is a pair* $(X, \mathscr{C})$ *where* X *is a set, and the* closure operator $\mathscr{C} : \wp(X) \to \wp(X)$ *assigns to each subset of* X *its* closure*, obeying to the following laws, for all* $A, B \subseteq X$*:*

1. $\mathscr{C}(\emptyset) = \emptyset$;
2. $A \subseteq \mathscr{C}(A)$;
3. $\mathscr{C}(A \cup B) = \mathscr{C}(A) \cup \mathscr{C}(B)$.

A closure space may be derived starting from a *binary relation*, that is, a *graph*. Closure spaces obtained this way are also known as *quasi-discrete closure spaces*. An example is shown in Fig. 2.

In [16] the authors provided a logical operator corresponding to the closure operator on sets of points in space, and a spatial operator inspired by the temporal *until* operator, fundamental in the classical temporal setting, arriving at the definition of a logic which is able to describe unbounded areas of space. Intuitively, the spatial until operator, which in this work we call *surrounded*, describes a situation in which it is not possible to "escape" an area of points satisfying a certain property, unless by passing through at least one point that satisfies another given formula. This operator was also inspired by a similar operator for *topological* spaces discussed by Aiello and van Benthem in [1, 7]. In [16] the authors also presented a model-checking algorithm for SLCS when interpreted on finite models. Assume a finite or countable set *AP* of *atomic propositions*. The syntax of SLCS is defined by the grammar in Fig. 1, where a ranges over *AP*, $\top$ denotes the truth value *true*, $\neg$ is negation, $\wedge$ is conjunction, $\mathscr{N}$ is the *near* operator and $\mathscr{S}$ is the *surrounded* operator.

More precisely, SLCS is equipped with two *spatial operators*: a "one step" modality, called "near" and denoted by $\mathscr{N}$, turning the closure operator $\mathscr{C}$ into a logical operator, and a binary *spatial until* operator $\mathscr{S}$,

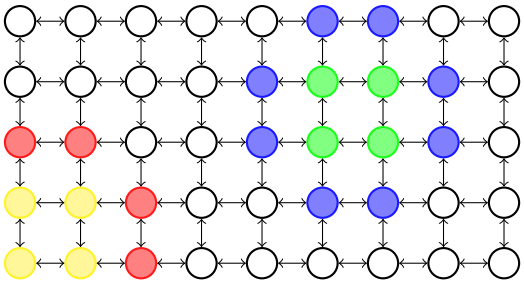


Figure 2: A graph inducing a *quasi-discrete* closure space

which is a spatial counterpart of the temporal *until* operator. The proposed spatial logic combines these new operators with standard boolean operators. A large number of useful derived operators exist, among which a particular form of reachability defined as the dual of the surrounded operator $(\phi_1 \mathscr{R} \phi_2 \triangleq \neg((\neg\phi_2)\mathscr{S}(\neg\phi_1))$, the *from-to* operator $(\phi_1 \mathscr{T} \phi_2 \triangleq \phi_1 \wedge ((\phi_1 \vee \phi_2)\mathscr{R}\phi_2))$ and the well-known operators 'everywhere' and 'somewhere' which can be defined as $\mathscr{E}\phi \triangleq \phi \mathscr{S} \bot$ and $\mathscr{F}\phi \triangleq \neg\mathscr{E}(\neg\phi)$, respectively.

The informal interpretation of formulas is as follows; their formal definition can be found in [16]. Atomic propositions and boolean connectives have the expected meaning. A point x satisfies formula $\mathscr{N}\phi$ if x is an element of the closure of the set of points satisfying ϕ. For formulas of the form $\phi_1 \mathscr{S} \phi_2$, the basic idea is that point x satisfies $\phi_1 \mathscr{S} \phi_2$ whenever there is "no way out" from ϕ_1 except passing by a point that satisfies ϕ_2. For instance, if we consider the model of Figure 2, *yellow* nodes satisfy $yellow \mathscr{S} red$ while *green* nodes satisfy $green \mathscr{S} blue$.

A point x satisfies $\phi_1 \mathscr{R} \phi_2$ if and only if either ϕ_2 is satisfied by x or there exists a sequence of points after x, all satisfying ϕ_1, leading to a point satisfying both ϕ_2 and ϕ_1. In the second case, it is not required that x itself satisfies ϕ_1. For instance, both *red* and *green* nodes in Figure 2 satisfy $(white \vee blue)\mathscr{R} blue$, as well as the white and blue nodes. The formula is not satisfied by the yellow nodes. This is so because the first node of a path leading to a blue node is not required to satisfy white or blue. It is easy to strengthen the notion of reachability when we want to identify all white nodes from which a blue node can be reached by requiring in addition that the first node of the path has to be white. This is the reason for the introduction of derived operator $\mathscr{T}$. Note that ϕ_2 is occurring also in the first argument of $\mathscr{R}$ in the definition of $\mathscr{T}$. This is because satisfaction of $\phi_1 \mathscr{R} \phi_2$ requires that the final node on the path satisfies both ϕ_1 and ϕ_2. We show further examples of the use of $\mathscr{T}$ shortly.

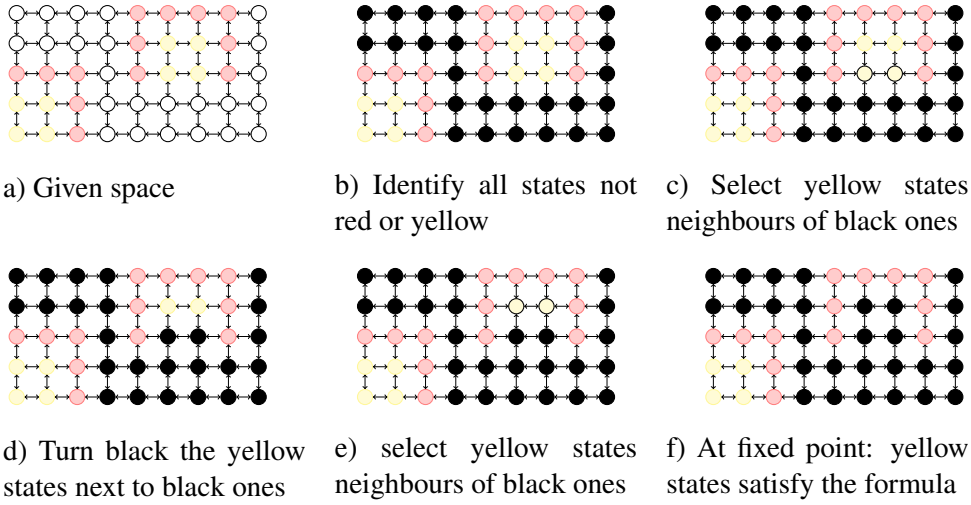
The spatial model-checking algorithms described in [16] are available as a proof-of-concept tool[1]. The tool is implemented in OCaml[2], and can be invoked as a global model checker for SLCS. The time complexity of the spatial model checking algorithm is *linear* in the number of points and arcs in the space and in the size of the formula. The algorithm for the surround operator is a global flooding algorithm illustrated in an informal way in Fig. 3 for the formula $Yellow \mathscr{S} Red$. First all points that certainly do not satisfy the formula (i.e. those that are neither *Yellow* nor *Red*) are marked black, then yellow points that are neighbours of black ones are removed from the set of points that potentially satisfy the formula by turning them black in successive steps until a fixed point is reached. The remaining yellow points satisfy the formula. The actual algorithm incorporates several optimisations.

As an illustration, in the following we show the application of the spatial model checker on a digital image representing a maze (see Fig. 4). The image is the (quasi discrete) closure space generated by the set of pixels (i.e. points of the closure space) and their adjacency relation, forming a regular graph, much like in Fig. 2. The green area is the exit. The blue areas (rectangular and round spots) are starting points.

The spatial model checker can be used to identify (sets of) points (by marking them with a specific colour of choice) that satisfy particular spatial properties. In this case three formulas are used to identify interesting areas. The formulas make indirect use of the $\mathscr{S}$ operator, by means of the derived operator $\mathscr{R}$ discussed earlier

[1] Web site: http://www.github.com/vincenzoml/topochecker.

[2] See http://ocaml.org.



a) Given space

b) Identify all states not red or yellow

c) Select yellow states neighbours of black ones

d) Turn black the yellow states next to black ones

e) select yellow states neighbours of black ones

f) At fixed point: yellow states satisfy the formula

Figure 3: Flooding algorithm evaluating $Yellow \mathcal{S} Red$

and further derived operators. In the following we are interested in three sets of points, each characterised by an *SLCS* formula.

1) White points from which an exit can be reached.

$$toExit = white \mathcal{T} green$$

The model checking result is shown in Fig. 5. Points that satisfy this formula are the yellow and orange ones.

2) Regions containing a starting point (blue area) from which an exit can be reached. These are white points from which it is both possible to reach an exit and to reach a blue point (starting point).

$$fromStartToExit = toExit \wedge (white \mathcal{T} blue)$$

Points that satisfy this formula are the orange ones in Fig. 5.

3) Points that are starting points and from which the exit can be reached. These are the blue points depicted as a rectangular shape for easy recognition in the image.

$$startCanExit = blue \mathcal{T} fromStartToExit$$

Points that satisfy this formula are shown in red in Fig. 5. These are indeed only the two rectangular shapes (coloured red as result) and not the two round shapes (that remained blue), since from the latter it is not possible to reach an exit.

The execution time of the spatial model checker on images consisting of about 1.0 mega pixels is in the order of several seconds on a standard laptop equipped with a 2Ghz processor.

The prototype spatial model checker has also been successfully applied to the bus transportation case study of the project [17]. In particular, to detect problems in the automatic vehicle location (AVL) data that is provided as input to other systems that in turn provide information to passengers and system operators such as bus arrival prediction systems. Such data may contain errors originating in a problem with the hardware of the measurement device or also indicate operational problems experienced by bus drivers that encountered unexpected road works or accidents and have to deviate from their planned route. The GPS data of the position of the buses can be automatically projected onto a portion of the digital map (such as for example provided

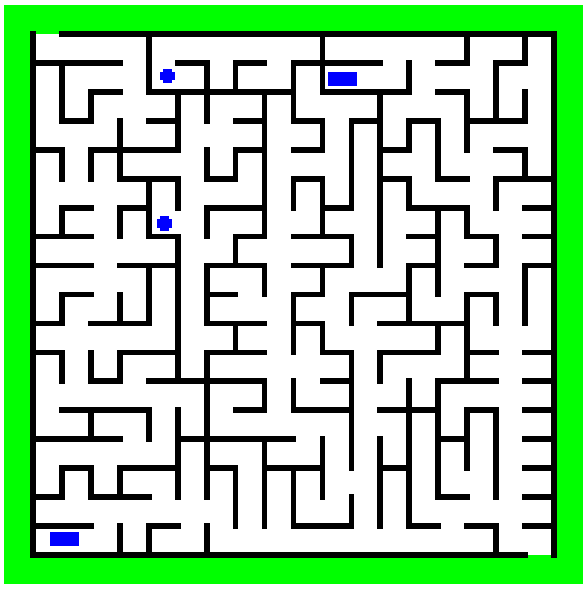


Figure 4: A maze.

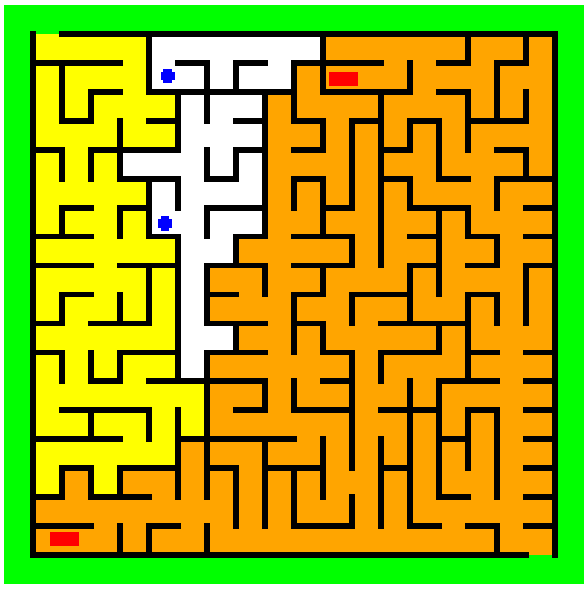


Figure 5: Model checker output.

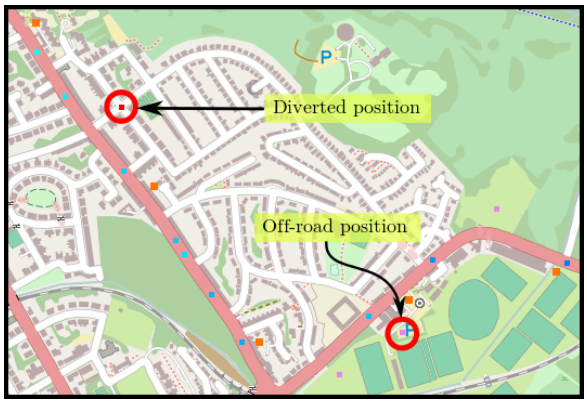


Figure 6: Diverted positions (neither off road, nor on a main street) are automatically highlighted in red by the spatial model checking procedure, off-road positions are highlighted in violet.

by OpenStreetMap[3]) of the area where the bus is operating, after which spatial model checking can be applied on an image of the map to identify those bus positions that may indicate a problem. An example of buses apparently being diverted or in an off-road position is shown in Fig. 6. Being able to detect and identify errors in AVL data, associating them with the right category of error in an automatic way, is of great use in obtaining better prediction algorithms. More complicated examples are those related to operational errors. For example one can detect buses of the same route that have overtaken each other or that are approaching the same bus stop within a too short interval of time.

3 Adding Time to the Spatial Logic for Closure Spaces

Starting from a spatial formalism and from a temporal formalism, *spatio-temporal* logics may be defined, by introducing some mutually recursive nesting of spatial and temporal operators. Several combinations can be obtained, depending on the chosen spatial and temporal fragments, and the permitted forms of nesting of the two. A large number of possibilities are explored in [24], for spatial logics based on topological spaces. One

[3]https://www.openstreetmap.org.

$$
\begin{array}{rcll}
\Phi & ::= & \top & [\textsc{True}] \\
 & | & p & [\textsc{Atomic predicate}] \\
 & | & \neg\Phi & [\textsc{Not}] \\
 & | & \Phi \vee \Phi & [\textsc{Or}] \\
 & | & \mathscr{N}\,\Phi & [\textsc{Close}] \\
 & | & \Phi\,\mathscr{S}\,\Phi & [\textsc{Surrounded}] \\
 & | & \mathtt{A}\,\varphi & [\textsc{All Futures}] \\
 & | & \mathtt{E}\,\varphi & [\textsc{Some Future}] \\
 & & & \\
\varphi & ::= & \mathscr{X}\,\Phi & [\textsc{Next}] \\
 & | & \Phi\,\mathscr{U}\,\Phi & [\textsc{Until}]
\end{array}
$$

Figure 7: STLCS syntax

such structure was investigated in Task 3.1, in the setting of closure spaces, namely the combination of the temporal logic *Computation Tree Logic* (CTL) with SLCS, resulting in the *Spatio-Temporal Logic of Closure Spaces* (STLCS). In STLCS spatial and temporal fragments may be arbitrary and mutually nested. As a proof of concept, a model checking algorithm has been defined, which is a variant of the classical CTL labelling algorithm [19, 2], augmented with the algorithm in [16] for the spatial fragment. The algorithm, which operates on finite spaces, has been implemented as a prototype [23], which is described in [13]. The same algorithm is also implemented in the tool `topochecker` [12] and was used in [14] in order to check spatio-temporal properties of bike sharing systems.

STLCS is interpreted on a variant of Kripke models, where valuations are interpreted at points of a closure space. Fix a set AP of proposition letters. STLCS formulas are defined by the grammar shown in Fig. 7, where p ranges over AP. The logic features the CTL path quantifiers `A` ("for all paths"), and `E` ("there exists a path"). As in CTL, such quantifiers must necessarily be followed by one of the path-specific temporal operators, such as[4] $\mathscr{X}\Phi$ ("next"), $\mathtt{F}\Phi$ ("eventually"), $\mathtt{G}\Phi$ ("globally"), $\Phi_1\,\mathscr{U}\,\Phi_2$ ("until"), but unlike CTL, in this case Φ, Φ_1 and Φ_2 are STLCS formulas that may make use of spatial operators, e.g. $\mathscr{N}$, $\mathscr{S}$ and operators derived thereof (see Sect. 2.)

The mutual nesting of such operators permits one to express spatial properties in which the involved points are constrained to certain temporal behaviours. Let us proceed with a few examples. Consider the STLCS formula `EG (green S blue)`. This formula is satisfied in a point x in the graph, associated to state s, if there exists a (possible) evolution of the system (i.e. path in the computation), starting from s, in which point x, in every state in the path, satisfies *green* and is surrounded by blue. A further, nested, example is the STLCS formula `EF (green S (AX blue))`. This formula is satisfied by a point x in the graph, in state s, if there is a (possible) evolution of the system, starting from s, in which point x is eventually green and is surrounded by points y that, for every possible evolution of the system from then on, will be blue in the next time step.

The complexity of the implemented algorithm is linear in the product of three quantities: 1) the sum of the number of states and the number of transitions of the temporal model; 2) the size of the formula; 3) the sum of the number of points and the number of arcs of the space. Such efficiency is sufficient for experimenting with the logic, and it is comparable to the efficiency of classical in-memory model chcking algorithms when the spatial part of the model is relatively small, but there is ongoing work towards further improvements. The algorithm has been implemented as an extension of the spatial model checker discussed in the previous section and is available as a proof-of-concept tool[5].

The tool has been applied (in its various implemented versions) on two case studies of the project. In ongoing work further properties of vehicular movement in public transport systems have been analysed. In particular a phenomenon known as *clumping*, which may occur in so-called "frequent" services – those where

[4]Some operators may be derived from others; for this reason in Fig. 7 we use a minimal set of connectives. As usual in logics, there are several different choices for such a set.

[5]Web site: http://www.github.com/vincenzoml/topochecker.

a timetable is not published. Clumping occurs where one bus catches up with – or at least comes too close to – the bus which is in front of it. In the absence of a published timetable for frequent services the important performance metric to consider is not timetable adherence but headway, a measure of the separation between subsequent buses. This separation can be defined both in terms of distance between buses on the same route or in terms of the time between two buses on the same route passing by the same bus stop. It is possible to identify these two notions of clumping using STLCS on a time series of street map images on which the bus positions are projected in a similar way as shown in Fig. 6. An impression of the results is shown in Fig. 8. This kind of analysis is particularly interesting to study the effectiveness of adaptive strategies that are proposed to avoid or mitigate the clumping phenomenon, providing better service to the public.

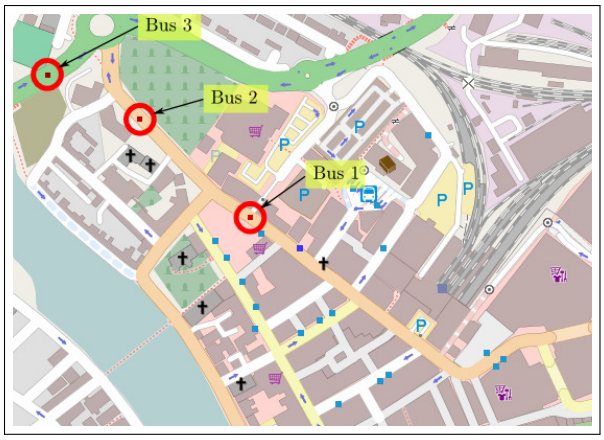
In [14] spatio-temporal model checking has been used to detect the emergent formation of 'clusters' of full (and empty) stations in the simulation traces of a Markov Renewal Process (MRP) model of large bike sharing systems [30], such as the one in London. The model describes the dynamics of a population of rational agents, coupled with the dynamics of bicycle stations in a two-dimensional rectangle representing a city. Interestingly, the simulation traces of the MRP model show a good correspondence with observed cycling times in the real system (see Fig. 9). The cycling time distribution shows a curious feature, namely the tail of the distribution is algebraic. This indicates that, contrary to the expectation, there are relatively many cases in which bike users return their bikes after much longer time than could be expected given the 30 minutes free allowance, and do so in a sort of 'accidental' or 'unforeseen' way. The hypothesis is that these late returns could be explained by the problems that users experience in returning their bikes when no parking places are available close to their desired destination. A model in which users are moving randomly from one place to another, using the bike sharing system, but always with the aim to return their bike within 30 minutes to a station, shows much fewer users with a late return. However, to reproduce the heavier algebraic tail it turned out to be sufficient to 'stress' the bike sharing system by introducing flows of commuters. Using spatio-temporal model checking it was shown that the introduction of the flows also leads to the emergence of clusters of full stations which are a plausible explanation for later-than-expected returns. Clusters were not detected in traces obtained from the random model without flows.

The spatial structure used in the MRP model of the BSS is a regular grid where nodes represent stations. Each station is labelled with information about how many bikes and free places are available. Simulation traces of the MRP model consist of a series of such grids where in every state the information about the stations is updated. In STLCS we can define the property `nextCluster` which characterises those points that will eventually become full and, for every future state, whenever full, they will stay full until they become part of a cluster. A cluster is defined as a full station in which all its four direct neighbours are full as well. `nextCluster` is a strong property that few points (stations) possess. Indeed, such points are central in cluster formation, as they represent stations that always form a cluster when they become full. In Fig. 10, the stations that satisfy this property are indicated in red. They are shown with reference to a state of the simulation trace where there are relatively many of them. For comparison the boundary of the set of points that will become a cluster, that is, those points satisfying $(\texttt{!EF cluster}) \,\&\, (\texttt{N EF cluster})$ are indicated in green.

4 Adding Space to the Signal Temporal Logic

Signal Spatio-Temporal Logic (SSTL) is an extension of Signal Temporal Logic [20, 29] with two spatial modalities. The first one, the *bounded somewhere* operator $\Diamond_{[w_1,w_2]}$ is taken from [33], while the second one, the *bounded surround* operator $\mathscr{S}_{[w_1,w_2]}$, is inspired by the *Spatial Logic for Closure Spaces* [16]. The logic comes with a boolean and quantitative semantics which can be found in [33, 32]. The boolean semantics defines when a formula is satisfied, the quantitative semantics provides and indication of the robustness with which a formula is satisfied [3, 5], i.e. how susceptible it is to changing its truth value for example as a result of a perturbation in the signals.

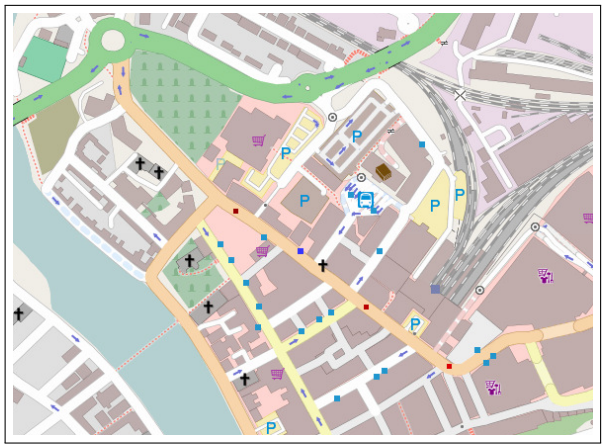
(a) Initial state



(b) Second state

(c) Bus 1 passes by the stop

(d) Bus 2 passes by the stop



(e) Final state

(f) Result from the model-checker. Points of the initial state that will be involved in a future conglomerate are coloured in green and indicated by the red circle

Figure 8: Detection of spatio-temporal conglomerate when bus 1 (c) and bus 2 (d) pass by the same bus stop (a) within a very short time period.

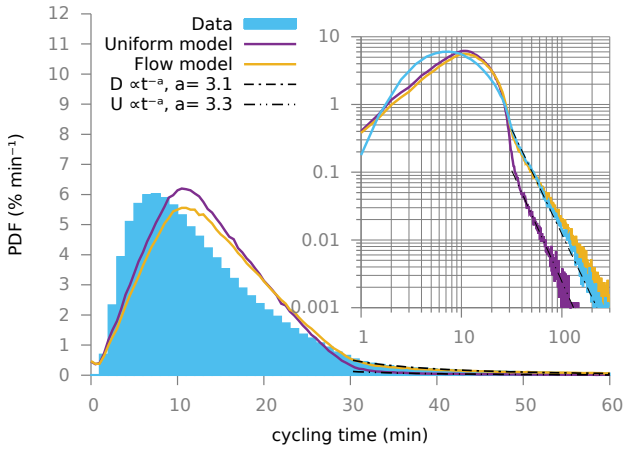


Figure 9: Cycling duration histograms (Data) in London, using 831,754 trip records in October 2012, and results of simulation of the uniform model (dark lines) and the flow model (light lines). Maintenance trips are not considered.

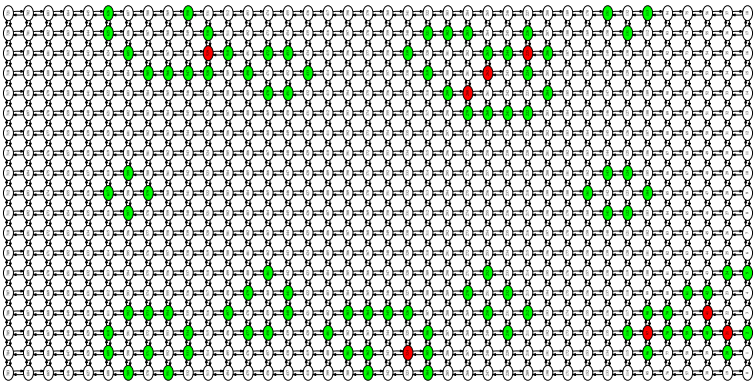


Figure 10: Stations of state 80 of the simulation that are on the boundary of the region of points that will eventually become a cluster (green), and stations that, whenever they are full, stay full until the end of the simulation trace and become part of a cluster (red).

4.1 Signal Spatio-Temporal Logic

SSTL is interpreted on spatio-temporal, real-valued signals. Space is discrete and described by a weighted graph $G = (L, E, w)$, while the time domain $\mathbb{T}$ will usually be the real-valued interval $[0, T]$, for some $T > 0$. In the following, we first introduce spatio-temporal signals, and then present the syntax of SSTL. A spatio-temporal trace is a function $\mathbf{x} : \mathbb{T} \times L \to \mathbb{D}$, where $\mathbb{D} \subseteq \mathbb{R}^n$ is the codomain of the trace. As for temporal traces, we write $\mathbf{x}(t,\ell) = (x_1(t,\ell), \cdots, x_n(t,\ell)) \in \mathbb{D}$, where each $x_i : \mathbb{T} \times L \to \mathbb{D}_i$, for $i = 1, ..., n$, is the projection on the i^{th} coordinate/variable.

Spatio-temporal traces can be obtained by simulating a stochastic model or a deterministic model, i.e. specified by a set of differential equations. In [33] the framework of patch-based population models is discussed, which generalise population models and are a natural setting from which both stochastic and deterministic spatio-temporal traces of the considered type emerge. An alternative source of traces are measurements of real systems.

Spatio-temporal traces are converted into spatio-temporal boolean or quantitative signals. Similarly to the case of STL, each *atomic predicate* μ_j is of the form $\mu_j(x_1, \ldots, x_n) \equiv (f_j(x_1, \ldots, x_n) \geq 0)$, for $f_j : \mathbb{D} \to \mathbb{R}$. Each atomic proposition gives rise to a spatio-temporal signal. In the boolean case, one may define function $s_j : \mathbb{T} \times L \to \mathbb{B}$; given a trace $\mathbf{x}$, this gives rise to the boolean signal $s_j(t,\ell) = \mu_j(\mathbf{x}(t,\ell))$ by point-wise lifting.

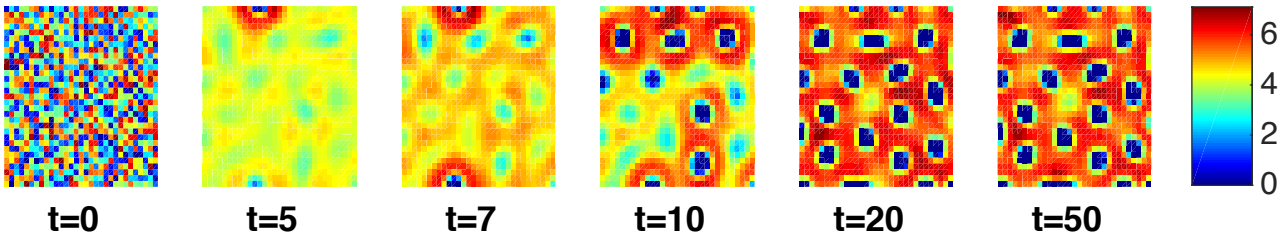


Figure 11: Value of x^A for the Turing system in [32] for $t = 0,5,7,12,20,50$ time units with parameters $K = 32, R_1 = 1, R_2 = -12, R_3 = -1, R_4 = 16, D_1 = 5.6$ and $D_2 = 25.5$. The initial condition has been set randomly. The colour map for the concentration is specified in the legend on the right.

Similarly, a quantitative signal is obtained as the real-valued function $s_j : \mathbb{T} \times L \to \mathbb{R}$, with $s_j(t,\ell) = f_j(\mathbf{x}(t,\ell))$. When the space L is finite a spatio-temporal signal can be presented as a finite collection of temporal signals.

The syntax of SSTL is given by

$$\varphi := \mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathscr{U}_{[t_1,t_2]}\, \varphi_2 \mid \; \diamondsuit_{[w_1,w_2]}\varphi \mid \varphi_1 \,\mathscr{S}_{[w_1,w_2]}\varphi_2.$$

Atomic predicates, boolean operators, and the until operator $\mathscr{U}_{[t_1,t_2]}$ are those of STL. The spatial operators are the *somewhere* operator, $\diamondsuit_{[w_1,w_2]}$, and the *bounded surround* operator $\mathscr{S}_{[w_1,w_2]}$, where $[w_1,w_2]$ is a closed real interval with $w_1 < w_2$. The spatial somewhere operator $\diamondsuit_{[w_1,w_2]}\varphi$ requires φ to hold in a location reachable from the current one with a total cost greater than or equal to w_1 and lesser than or equal to w_2. The surround formula $\varphi_1 \,\mathscr{S}_{[w_1,w_2]}\varphi_2$ is true in a location ℓ, for the trace $\mathbf{x}$, when ℓ belongs to a set of locations A satisfying φ_1, such that its external boundary $B^+(A)$ (i.e., all the nearest neighbours external to A of locations in A) contain only locations satisfying φ_2. Furthermore, locations in $B^+(A)$ must be reached from ℓ by a shortest path of cost between w_1 and w_2. Hence, the surround operator expresses the topological notion of being surrounded by a φ_2-region, with additional metric contraints. We can also derive the *everywhere* operator $\boxdot_{[w_1,w_2]}\varphi := \neg\diamondsuit_{[w_1,w_2]}\neg\varphi$ requiring φ to hold in all the locations reachable from the current one with a total cost between w_1 and w_2.

SSTL can be used to identify the formation of *patterns* in a reaction-diffusion system. From the point of view of formal verification, the formation of patterns is an inherently spatio-temporal phenomenon, in that the relevant aspect is how the spatial organisation of the system changes over time. Alan Turing theorised in [40] that pattern formation is a consequence of the coupling of reaction and diffusion phenomena involving two different chemical species, and can be described by a set of PDE reaction-diffusion equations, one for each species. Fig. 11 shows an example simulation of such a PDE for a selected point in time, starting from a random spatial distribution of the concentration of the two chemical species.

Spots with a low concentration of species A can be identified by points in space satisfying the following SSTL formula where x^A denotes the concentration of species A:

$$\phi_{\text{spot}} := (x^A \leq h)\,\mathscr{S}_{[w_1,w_2]}(x^A > h). \tag{1}$$

The use of distance bounds w_1 and w_2 in the surround operator allows one to constrain the size/diameter of the spot to $[w_1,w_2]$. To identify the insurgence time of the pattern and whether it remains stable over time the spatial property needs to be combined with temporal operators in the following way:

$$\phi_{\text{pattern}} := \mathscr{F}_{[T_{\text{pattern}},T_{\text{pattern}}+\delta]}\mathscr{G}_{[0,T_{\text{end}}]}(\phi_{\text{spot}}); \tag{2}$$

ϕ_{pattern} means that eventually ($\mathscr{F}$) at a time between T_{pattern} and $T_{\text{pattern}} + \delta$ the property spot becomes true and remains true ($\mathscr{G}$) for at least T_{end} time units. Fig. 12(b) shows the validity of the property ϕ_{pattern} in each cell $(i,j) \in L$, for both the boolean and the quantitative semantics.

4.2 Robustness Analysis and System Design

In [4], the authors combined SSTL with recent machine learning approaches to verification [9] and synthesis [3, 5], to analyse the robustness of stochastic spatio-temporal models and to design specific behaviours.

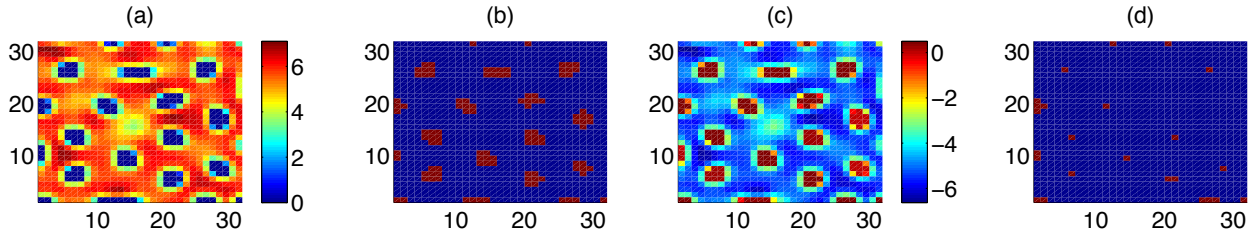


Figure 12: Validity of formula (2) with parameters $h = 0.5, T_{\text{pattern}} = 19, \delta = 1, T_{\text{end}} = 30, w_1 = 1, w_2 = 6$ for (b), (c) and $w_2 = 4$ for (d). (a) Concentration of A at time t = 50; (b) (d) Boolean semantics of the property ϕ_{pattern}; the cells (locations) that satisfy the formula are in red, the others are in blue; (c) Quantitative semantics of the property ϕ_{pattern}; The value of the robustness is given by a colour map as specified in the legend on the right of the figure.

The idea is to extend the semantics of SSTL to stochastic systems along the lines of [3, 5], by considering either the probability with which a formula is satisfied by a stochastic model, for the qualitative semantics, or the distribution of the robustness score, for the quantitative one. This allows to combine SSTL with statistical model checking tools. In particular, in [4] the authors consider a model of developmental biology, the french flag pattern formation, in which the spatial gradient of the Bicoid protein is responsible for the horizontal segmentation of the *Drosophila* embryo. Two different kinds of analysis are discussed in [4]

1. The robustness of the property encoding the french flag property (i.e. a decaying concentration of the signalling protein along the horizontal axis, identifying three regions, of high, medium and low expression), is studied with respect to the model parameters. To this end, the authors exploit a novel method, smoothed model checking [9], which allows for the statistical reconstruction of the satisfaction probability as a function of model parameters;

2. The model is optimally designed in order to maximise the robustness of the SSTL behavioral specification. The authors here use the statistical system design approach presented in [3, 5], which leverages a recent provably convergent Bayesian optimisation algorithm.

Moreover, the Java implementation of SSTL is being integrated in the tool U-check [10], which implements these novel statistical verification techniques.

5 Other Approaches to Spatio-temporal Performance Evaluation

Several other approaches have been explored to the spatial and spatio-temporal analysis of CAS. In this section we describe two of these approaches. The work in [21] studies several spatial and spatio-temporal performance measures for spatially distributed CAS, whereas the second approach develops a reaction-diffusion partial-differential approximation for spatial stochastic process algebra that leads to a computationally more efficient solution of spatial models which include mobility of objects [37].

5.1 Performance Measures in Spatially Distributed CAS

In [21] the authors explored what performance measures would look like in a spatially distributed CAS. In classical computer systems, the trajectory of the system consists of pairs (s,t) where s records the state of the system, and t the time at which that state is entered. Performance measures are typically defined over a temporal sequence of states, or averaged over a steady state behaviour when the probability distribution over states is no longer changing, meaning that the time aspect can be abstracted away. But when we consider a spatially distributed component the trajectory of the component will consist of triples (s,ℓ,t), where s and t will be behavioural state and time as previously, and ℓ is the current location of the component. For a collective system we must take into consideration such a trajectory component for each system component, leading to a

trajectory in which each time entry has a triple for each component, or a trajectory in which each timed entry is a vector of populations $(\mathbf{v},t)$, where $\mathbf{v}$ is a vector with one entry for each possible state, location pair and $\mathbf{v}(s,\ell)(t)$ records the number of components in state s at location ℓ at time t.

In [21] the authors follow this latter approach and consider the performance measures which might be defined considering the trajectories with respect to space as well as with respect to time. Thus for single agent A we might consider measures such as the times at which a given location was visited $visit(A,l)$, or the locations that were visited at a given time $loc(A,t)$. Considering notions of steady state we then have a probability distribution over locations, or cumulative probability distribution over time with respect to a specific location. Using these measures as primitive a number of novel spatio-temporal measures were derived for a simple model of movement in a group of robots captured in PALOMA. For example, Figure 13, shows the probability of the leader being present in different locations at time 100s, given that it started in location $(30,30)$ at time 0.

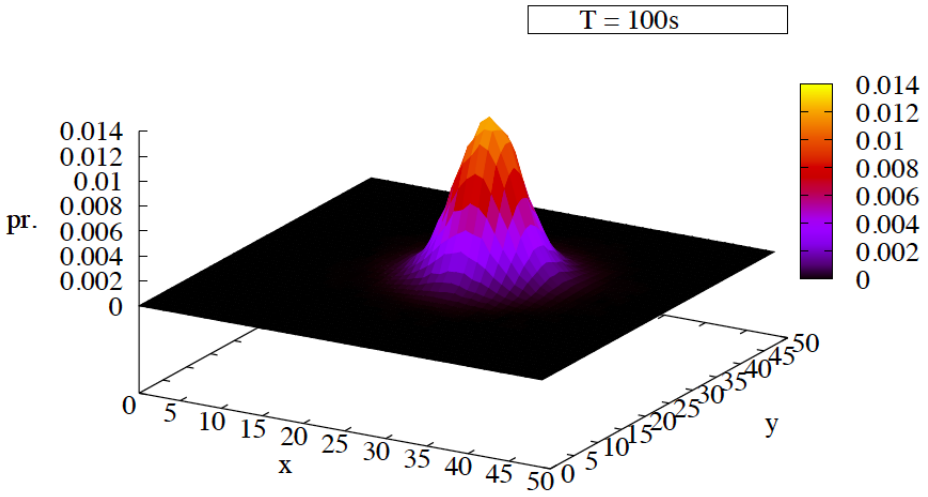


Figure 13: The spatial probability distribution of the leader of a group of robots at time 100s, given an initial position of $(30,30)$ at time 0

5.2 A Partial-differential Approximation for Spatial Stochastic Process Algebra

In [37] the authors study CAS by means of a process algebra with ordinary differential equation (ODE) semantics featuring an explicit mobility model over a 2D lattice where processes may migrate to neighbouring regions independently, and interact with each other when they are in the same region. The underlying ODE system size grows linearly with the number of regions, hindering the analysis in practice. Assuming an unbiased random walk, the authors introduce an approximation in terms of a system of reaction-diffusion partial differential equations, of size independent of the lattice granularity. This is an approximate view of the model which replaces discrete movements across regions in a continuous fashion. The authors work with the process algebra FEPA (Fluid Extended Process Algebra) presented in [38, 39]. However, with appropriate changes the ideas are applicable to languages such as Bio-PEPA [18], Cardelli's Chemical Ground Form [11], and the continuous π-calculus [25].

In the literature concerned with models that are not derived from a process algebra, such PDE approximations are typically presented by first considering a static version of the system in terms of an ODE model, where locality is not explicitly taken into account. Such a static description defines the model of local interactions. Then, a spatial domain and a mobility model are added, leading to a mobile model in terms of a PDE where a diffusive term captures movement, and a reactive term captures the interacting behaviour of the static model (cf., e.g., [34]). The authors proceed in [37] in an analogous fashion. The generic behaviour within any region is captured by some static FEPA model M, where space is not modelled. Then, a spatial lifting of the model

	$d=1$				$d=3$				$d=5$			
K	μ_1	*Time*	μ_2	*Time*	μ_1	*Time*	μ_2	*Time*	μ_1	*Time*	μ_2	*Time*
4	0.082	0 s	0.057	0 s	0.086	0 s	0.060	0 s	0.090	0 s	0.064	1 s
8	0.026	0 s	0.017	0 s	0.030	3 s	0.019	3 s	0.034	11 s	0.023	12 s
16	0.014	5 s	0.008	6 s	0.018	68 s	0.015	71 s	0.021	241 s	0.015	264 s
20	0.013	13 s	0.007	14 s	0.017	195 s	0.011	211 s	0.020	764 s	0.014	837 s
PDE Time		0 s		0 s		1 s		1 s		2 s		2 s

Table 1: Maximum absolute error between the ODE and PDE estimates of species S_1 at time 0.2 across the discrete mesh, for diffusion coefficients $\mu_1 = 0.10$ and $\mu_2 = 0.15$.

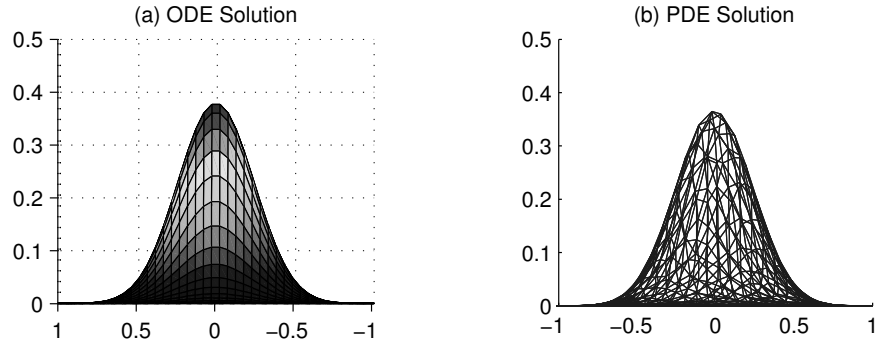


Figure 14: S_1 surfaces projected on the x-axis; $K = 20$, $d = 5$, $\mu = 0.15$.

is considered, $\mathscr{S}(M)$, which is a syntactic transformation which yields a FEPA model that makes the static version parametric with respect to the set of regions, and adds the random-walk behaviour. The PDE approximation is justified by observing that the ODE system of the spatial model $\mathscr{S}(M)$ can be written in terms of a discrete Laplace operator. The PDE is taken as the continuous approximation of such Laplacian.

At first sight, this approximation might appear of little use in practice. This is because, except for special cases, analytical PDE solutions are not available, and PDE systems are solved numerically using algorithms that discretise the continuous space [36]. However, when solving the spatial ODE system directly, the number of discrete regions is determined by the FEPA model, while the coarseness of the discrete mesh made by the PDE solver depends on the algorithm's parameters. Thus, the PDE solver may in effect perform a coarsening of the original spatial domain.

The approach is validated by considering a generalised Lotka-Volterra model [31, Section 3.2], where predators $S_1,\ldots,S_d$ prey on all the prey species $R_1,\ldots,R_d$ with different severities. In order to study the impact on the PDE system size (equal to $2d$) of the static version of the model, the authors considered the model for a different number of species, $d = 1,3,5$. To study the impact of discretisation, instead, the spatial domain $[-1;1]^2$ is discretised into uniform grids where regions are separated from each other by distance $1/K \in \{\frac{1}{4},\frac{1}{8},\frac{1}{16},\frac{1}{20}\}$, corresponding to increasingly finer discrete meshes with $(K-1)^2$ regions. In particular, the size of the underlying ODE system is $d\cdot(K-1)^2$. Also the impact of mobility on the model's dynamics is studied by considering different diffusion coefficients $\mu_{S_i} = \mu_{R_i} = \mu \in \{0.10, 0.15\}$ for all $1 \le i \le d$.

Denoting the 2D Laplace operator by $\triangle = \partial_{xx} + \partial_{yy}$, the corresponding PDE system is given by

$$\partial_t \omega_{S_i} = \omega_{S_i} \sum_{j=1}^{d} q_{i,j} r_{i,j} \omega_{R_j} - s_i \omega_{S_i} + \mu_{S_i} \triangle \omega_{S_i} \qquad \partial_t \omega_{R_j} = z_j \omega_{R_j} - \omega_{R_j} \sum_{i=1}^{d} r_{i,j} \omega_{S_i} + \mu_{R_j} \triangle \omega_{R_j}$$

$$\omega^0_{S_i}(x,y) := \begin{cases} 0 & ,x^2+y^2 \geq \frac{1}{4} \\ \frac{\exp(4)}{\exp\left(1/(\frac{1}{4}-(x^2+y^2))\right)} & ,\text{otherwise} \end{cases} \qquad \omega^0_{R_i} := 1 - \omega^0_{S_i}$$

where $1 \leq i, j \leq d$ (and $q_{i,j}, r_{i,j}, s_i, z_i$ are certain model parameters). The initial conditions $\omega^0_{S_i}$ enjoy a "bell shape" and are centered at $(0,0)$ with peak 1.0. Being given by $\omega^0_{R_i} = 1 - \omega^0_{S_i}$, instead, the initial conditions $\omega^0_{R_i}$ resemble the shape of a pit.

As a measure of the accuracy, the authors considered solutions for species S_1 at time $t = 0.20$. These were chosen arbitrarily, the latter ensuring that the solution was sufficiently far away from the initial condition. For each K, the error was defined as the maximum absolute difference across the whole spatial domain between the ODE solution at each point in the discrete mesh and the corresponding PDE solution (using linear interpolation to sample at the same coordinate). Matlab R2013b was used for the numerical solutions. The ODEs were solved using the built-in *ode15s* function, while the PDEs were solved using the function *parabolic* in the Partial Differential Equation Toolbox. All parameters were set as the default ones. Runtimes were taken on a machine with 16 GB RAM.

Table 1 presents the results, showing high quality of the approximation in general. Figure 14, instead, visualises the numerical solutions of S_1. The range of values attained by the ODE/PDE solutions were within $[0.00; 0.40]$, thus the absolute differences correspond to at most around 20% (for $K = 4$) relative to the peak values. The table shows higher accuracy with increasing K across all tests (cf. Figure 14 for a visual appreciation). However, PDEs are cheaper to solve than ODEs for fine meshes. In fact, the ODEs could not be solved for significantly larger values of K due to out-of-memory errors.

6 Scalable Model Checking

We briefly review additional work on scalable model checking. A more extended report will be provided in Deliverable 3.3 due in month 48. Work on model checking single agent behaviours by fluid approximation, with a detailed description of global fluid model checking algorithms and correctness proofs has been published in [8]. The theoretical foundations and algorithm for an on-the-fly fast mean field model checking procedure for the verification of single agents in the context of DTMC population models have been published in [27]. The algorithm has been implemented in the prototype model checker FlyFast, that was described in Deliverable 5.2. FlyFast has been used for modelling and analysis of an extended bike sharing model [6] briefly discussed in Deliverable 3.2 and is available for use on further case studies. It has furthermore been shown that, under certain conditions, FlyFast can be used to obtain approximate results of single agent behaviours in continuous time Markov population models, if the models are not too stiff [26]. FlyFast is based on an efficient on-the-fly probabilistic model checking approach described in [28]. Further work is foreseen in linking the model checking techniques to the modelling language CARMA developed in the context of WP4 and on extensions of the logic implemented in FlyFast.

7 Conclusions and Roadmap

Relationship to other work packages.

WP1 Emergent Behaviour and Adaptivity. Spatio-temporal model checking offers a complementary approach to the analysis and identification of emergent behaviour that evolves in a spatial context. Combined use of techniques developed in WP1 and WP3 would be interesting to explore further.

WP2 Collective Adaptive Behaviour in Space. One of the ideas that has been pursued in the work on spatial verification in WP3 is to develop a spatial logic with basic spatial operators that is general enough to be applied in the diverse spatial settings that were identified in the context of Work Package 2. Furthermore, metrics and distance functions can be added in an orthogonal way providing further spatial richness.

WP4 Language and Design Methodology. In WP3 the foundations have been developed for a variety of scalable verification techniques of which a number have been implemented as prototype tools. As a first step it can be expected that CARMA models can be used to generate stochastic or deterministic simulation traces that can be used for spatial model checking in much a similar way as spatio-temporal model checking was applied on simulation traces from the MRP bike sharing model discussed earlier. Further integration would be desirable and is part of ongoing research.

WP5 Model Validation and Tool Support. A number of prototype tools have been developed in WP3 that are available for use for model validation. Some of those, such as the FlyFast on-the-fly mean field model checker have already been linked to the QUANTICOL web-site, others such as the spatio-temporal model checker topochecker, will soon be added.

Roadmap for the final reporting period concerning Task 3.1 Good progress has been made in developing scalable verification for spatial logics and promising results have been obtained in a variety of application areas. As a matter of fact, this research has opened up a promising and exciting new area of research that could be developed in many different ways exceeding the remaining time available in the project. In this roadmap we provide a few limited ideas that we plan to pursue in the short term in the context of WP3. This would require an extension of the duration of Task 3.1, which was originally due to finish in month 36.

- In some cases it may be important to be able to specify spatial properties concerning *groups* of points in space rather than of individual points. For example, the property that agents associated to points in space are able to connect to one another and act as a group, or that they are located all together in a protected environment, or that they can share part of the same route to reach a common exit or goal. In all such situations, it is important to be able to predicate over spatial aspects, and possibly find methods to certify that a given collective adaptive system satisfies specific requirements in this respect. This is one of the lines of ongoing work on the development of suitable spatial and spatio-temporal logics in the context of the project [15].
- Although the spatio-temporal model checking algorithms do work fine, there is much scope for further optimisations that are specific of spatial models. One of the ideas is to exploit the gradual change that can be expected in the evolution of behaviour in a spatial setting. Another idea could be to exploit notions of spatial bisimulation and spatial aggregation to reduce the size of the space that needs to be verified. A third option is to use abstraction and refinement techniques in model checking [19] in order to verify a system at a given level of abstraction, deepening exploration only on a by-need basis.
- Further work on robustness analysis for SSTL is foreseen and includes the development of prototype tools making the approach more widely available for application on CAS.
- A further integration of spatio-temporal model checking and fluid or mean field model checking would be interesting as well.

References

[1] M. Aiello. “Spatial Reasoning: Theory and Practice”. PhD thesis. Institute of Logic, Language and Computation, University of Amsterdam, 2002.

[2] C. Baier and J. P. Katoen. *Principles of model checking*. MIT Press, 2008, pp. I–XVII, 1–975. ISBN: 978-0-262-02649-9.

[3] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. "On the Robustness of Temporal Properties for Stochastic Models". In: *HSB*. Vol. 125. EPTCS. 2013, pp. 3–19.

[4] E. Bartocci, L. Bortolussi, D. Milios, L. Nenzi, and G. Sanguinetti. "Studying Emergent Behaviours in Morphogenesis Using Signal Spatio-Temporal Logic". In: *HSB*. Vol. 9271. Lecture Notes in Computer Science. Springer, 2015, pp. 156–172.

[5] E. Bartocci, L. Bortolussi, L. Nenzi, and G. Sanguinetti. "System design of stochastic models using robustness of temporal properties". In: *Theor. Comput. Sci.* 587 (2015), pp. 3–25.

[6] M. H. ter Beek, S. Gnesi, D. Latella, and M. Massink. "Towards Automatic Decision Support for Bike-Sharing System Design". In: *SEFM Workshops*. Vol. 9509. Lecture Notes in Computer Science. Springer, 2015, pp. 266–280.

[7] J. van Benthem and G. Bezhanishvili. "Modal Logics of Space". In: *Handbook of Spatial Logics*. Springer, 2007, pp. 217–298.

[8] L. Bortolussi and J. Hillston. "Model checking single agent behaviours by fluid approximation". In: *Inf. Comput.* 242 (2015), pp. 183–226.

[9] L. Bortolussi, D. Milios, and G. Sanguinetti. "Smoothed model checking for uncertain Continuous-Time Markov Chains". In: *Information and Computation* (2016), pp. –. ISSN: 0890-5401. DOI: http://dx.doi.org/10.1016/j.ic.2016.01.004. URL: http://www.sciencedirect.com/science/article/pii/S0890540116000055.

[10] L. Bortolussi, D. Milios, and G. Sanguinetti. "U-Check: model checking and parameter synthesis under uncertainty". In: *Quantitative Evaluation of Systems (QEST)*. Vol. 9259. Springer International Publishing. Springer International Publishing, 2015, 89–104. DOI: 10.1007/978-3-319-22264-6_6.

[11] L. Cardelli. "On process rate semantics". In: *Theor. Comput. Sci.* 391 (3 2008), pp. 190–215. ISSN: 0304-3975.

[12] V. Ciancia. *topochecker - a topological model checker*. http://fmt.isti.cnr.it/topochecker – https://github.com/vincenzoml/topochecker. 2015.

[13] V. Ciancia, G. Grilletti, D. Latella, M. Loreti, and M. Massink. "An experimental spatio-temporal model checker". In: *Revised selected papers of the SEFM 2015 collocated workshops*. LNCS. Springer, 2015.

[14] V. Ciancia, D. Latella, M. Massink, and R. Paskauskas. "Exploring spatio-temporal properties of bike-sharing systems". In: *Proceedings of First International Workshop on Spatial and COllective PErvasive Computing Systems (SCOPES). Workshop at IEEE SASO 2015*. IEEE, 2015.

[15] V. Ciancia, D. Latella, M. Loreti, and M. Massink. *Model Checking Spatial Logics for Closure Spaces — Extended Version*. Tech. rep. TR-QC-03-2016. QUANTICOL, 2016. URL: http://blog.inf.ed.ac.uk/quanticol/technical-reports/.

[16] V. Ciancia, D. Latella, M. Loreti, and M. Massink. "Specifying and verifying properties of space". In: *Proceedings of 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014*. LNCS 8705. Springer, 2014, pp. 222–235.

[17] V. Ciancia, S. Gilmore, D. Latella, M. Loreti, and M. Massink. "Data Verification for Collective Adaptive Systems: Spatial Model-Checking of Vehicle Location Data". In: *SASO Workshops*. IEEE Computer Society, 2014, pp. 32–37.

[18] F. Ciocchetta and J. Hillston. "Bio-PEPA: A framework for the modelling and analysis of biological systems". In: *Theor. Comput. Sci.* 410.33–34 (2009), pp. 3065–3084.

[19] E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT Press, 2001. ISBN: 978-0-262-03270-4. URL: http://books.google.de/books?id=Nmc4wEaLXFEC.

[20] A. Donzé, T. Ferrer, and O. Maler. "Efficient Robust Monitoring for STL". In: *Proc. of CAV*. 2013.

[21] C. Feng, M. Gribaudo, and J. Hillston. "Performance Analysis of Collective Adaptive Behaviour in Time and Space". In: *Electr. Notes Theor. Comput. Sci.* 318 (2015), pp. 53–68.

[22] A. Galton. "A generalized topological view of motion in discrete space". In: *Theoretical Computer Science* 305.1–3 (2003), pp. 111 –134. ISSN: 0304-3975. DOI: http://dx.doi.org/10.1016/S0304-3975(02)00701-6. URL: http://www.sciencedirect.com/science/article/pii/S0304397502007016.

[23] G. Grilletti and V. Ciancia. *STLCS model checker*. https://github.com/cherosene/ctl_logic. 2014.

[24] R. Kontchakov, A. Kurucz, F. Wolter, and M. Zakharyaschev. "Spatial Logic + Temporal Logic = ?" In: *Handbook of Spatial Logics*. Ed. by M. Aiello, I. Pratt-Hartmann, and J. van Benthem. Springer, 2007, pp. 497–564. DOI: 10.1007/978-1-4020-5587-4_9. URL: http://dx.doi.org/10.1007/978-1-4020-5587-4_9.

[25] M. Kwiatkowski and I. Stark. "The continuous pi-calculus: A process algebra for biochemical modelling". In: *CMSB*. LNCS 5307. 2008, pp. 103–122.

[26] D. Latella, M. Loreti, and M. Massink. "On-the-fly Fluid Model Checking via Discrete Time Population Models". In: *EPEW*. Vol. 9272. Lecture Notes in Computer Science. Springer, 2015, pp. 193–207.

[27] D. Latella, M. Loreti, and M. Massink. "On-the-fly PCTL fast mean-field approximated model-checking for self-organising coordination". In: *Sci. Comput. Program.* 110 (2015), pp. 23–50.

[28] D. Latella, M. Loreti, and M. Massink. "On-the-fly Probabilistic Model Checking". In: *ICE*. Vol. 166. EPTCS. 2014, pp. 45–59.

[29] O. Maler and D. Nickovic. "Monitoring Temporal Properties of Continuous Signals". In: *Proc. FORMATS*. 2004.

[30] M. Massink and R. Paškauskas. "Model-based Assessment of Aspects of User-satisfaction in Bicycle Sharing Systems". In: *18th IEEE International Conference on Intelligent Transportation Systems*. Ed. by M. Sotelo Vazquez, C. Olaverri Monreal, J. Miller, and A. Broggi. DOI: 10.1109/ITSC.2015.224. IEEE. IEEE, 2015, pp. 1363–1370.

[31] J. D. Murray. *Mathematical Biology I: An Introduction*. 3rd. Springer, 2002.

[32] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink. "Qualitative and quantitative monitoring of spatio-temporal properties". In: *Proceedings of Runtime Verification 2015*. LNCS 9333. Springer, 2015, pp. 21–37.

[33] L. Nenzi and L. Bortolussi. "Specifying and Monitoring Properties of Stochastic Spatio-Temporal Systems in Signal Temporal Logic". In: *VALUETOOLS*. ICST, 2014.

[34] A. Okubo and S. A. Levin. *Diffusion and Ecological Problems : Modern Perspectives*. Interdisciplinary Applied Mathematics. Springer, 2001.

[35] M. B. Smyth and J. Webster. "Discrete Spatial Models". In: *Handbook of Spatial Logics*. Ed. by Springer. Springer, 2007, pp. 713–798.

[36] J. W. Thomas. *Numerical Partial Differential Equations: Finite Difference Methods*. Springer-Verlag, 1995.

[37] M. Tschaikowski and M. Tribastone. "A Partial-differential Approximation for Spatial Stochastic Process Algebra". In: *8th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2014, Bratislava, Slovakia, December 9-11, 2014*. Ed. by M. Haviv, W. J. Knottenbelt, L. Maggi, and D. Miorandi. ICST, 2014. DOI: 10.4108/icst.valuetools.2014.258170. URL: http://dx.doi.org/10.4108/icst.valuetools.2014.258170.

[38] M. Tschaikowski and M. Tribastone. "Exact fluid lumpability for Markovian process algebra". In: *CONCUR*. Vol. 7545. LNCS. 2012, pp. 380–394.

[39] M. Tschaikowski and M. Tribastone. “Extended Extended Differential Aggregations in Process Algebra for Performance and Biology”. In: *QAPL*. To appear. 2014.

[40] A. M. Turing. “The Chemical Basis of Morphogenesis”. In: *Philosophical Transactions of the Royal Society of London B: Biological Sciences* (1952).