Spatial Logic and Spatial Model Checking for Closure Spaces[⋆]

Vincenzo Ciancia[1], Diego Latella[1], Michele Loreti[2,3], and Mieke Massink[1]

[1] Istituto di Scienza e Tecnologie dell'Informazione 'A. Faedo', CNR, Pisa, Italy
[2] Università di Firenze, Italy
[3] IMT Alti Studi, Lucca, Italy

Abstract. Spatial aspects of computation are 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 concurrent systems; however, properties of space are typically not explicitly taken into account. This tutorial provides an introduction to recent work on a topology-inspired approach to formal verification of spatial properties depending upon (physical) space. A logic is presented, stemming from the tradition of topological interpretations of modal logics, dating back to earlier logicians such as Tarski, where modalities describe neighbourhood. These topological definitions are lifted to the more general setting of *closure spaces*, also encompassing discrete, graph-based structures. The present tutorial illustrates the extension of the framework with a spatial *surrounded* operator, leading to the spatial logic for closure spaces SLCS, and its combination with the temporal logic CTL, leading to STLCS. The interplay of space and time permits one to define complex spatio-temporal properties. Both for the spatial and the spatio-temporal fragment efficient model-checking algorithms have been developed and their use on a number of case studies and examples is illustrated.

1 Introduction

Modal logics, model checking and static analysis enjoy an outstanding mathematical tradition, spanning over logics, abstract mathematics, artificial intelligence, theory of computation, system modelling, and optimisation. However, the *spatial* aspects of systems, that is, dealing with properties of entities that relate to their position, distance, connectivity and reachability in *space*, have never been truly emphasised in computer science. With the recent interest in the design of fully *decentralised* systems that are composed of a large number of locally interacting objects that are distributed in physical space, also called *Collective Adaptive Systems* (CAS) [2], spatial reasoning and formal spatial verification have gained renewed interest. A starting point is provided by so-called *spatial logics* [1], that have been studied from the point of view of (mostly modal) logics. The field of spatial logics is well developed in terms of descriptive languages and

⋆ Research partially funded by EU project QUANTICOL (nr. 600708) aspects such as computability and complexity. The development dates back to work by early logicians such as Tarski, who studied possible semantics of classic modal logics, using topological spaces in place of Kripke frames. However, the frontier of current research does not yet address formal verification problems, and in particular, discrete spatial models are still a relatively unexplored field.

In this tutorial, we review some relevant current literature dealing with models where space is continuous, and start an analysis of the situation in the case of discrete structures. The interest in such an analysis comes from the conjecture that properties may be described using the same languages in the continuous, discrete, and relational (classical) case. The longer term aim is to provide a unifying view of temporal and spatial properties which is independent of the kind of models that are taken into account. The tutorial is intended to be a starting point for understanding which descriptive languages are most suitable for such an endeavour. As a next step we show how to cast the well known developments in spatio-temporal reasoning in the realm of discrete and finite structures, and develop efficient and effective verification algorithms. Their use will be illustrated on small and larger examples in the field of CAS. This development constitutes a novel, relatively unexplored research line. The lack of applications of spatial logic in the field of verification is also witnessed in the introduction to the Handbook of Spatial Logics [1], that we use as one of our main references. The tutorial is furthermore partially based on some of our previous and forthcoming joint work with various other authors (see [17,18,15,38,35,20]).

We can distinguish broadly three main aspects in this field. First of all, the *spatial structures* with which relevant aspects of space can be modelled must be identified. Second, suitable *spatial and spatio-temporal logics* must be developed. Finally it is useful to enrich the setting to be able to take *quantitative aspects* into account such as distance or probability. Let us briefly review these three aspects:

Spatial structures. Space can be modelled as a discrete or continuous entity. This ought to be accommodated in a general setting by choosing appropriate abstract mathematical structures. *Topological spaces* are typical examples. However, we shall see that for dealing with discrete spatial structures *closure spaces*, a generalisation of topological spaces, are a better starting point.

Spatial logics. Spatial logics predicate on properties of entities located in the space; for example, one may be interested in entities that are *inside*, *outside* or on the *boundary* of regions of space where certain properties hold. Depending on the specific logical language, the entities described can be:

- Points in the space. In this case reasoning has a strongly local flavour. Global properties (e.g., a region of a space not having “holes”) cannot be expressed.
- Spaces as a whole. Global properties can easily be expressed if the point of view is shifted from the behaviour of an individual in a specified setting, to the analysis of several possible global scenarios consisting of all the entities in a given space.

– Regions of space. This approach combines reasoning on multiple entities simultaneously with a focus on the interaction (e.g., overlapping or contact) between areas having different properties.

Spatio-temporal logics. The combination of spatial logics with other modal logics, such as temporal logics, provides an even richer language for the formulation of properties that reflect both spatial and temporal aspects at the same time. The combination of spatial and temporal logics introduces more design variables, especially for what concerns the interplay between the spatial and the temporal component. Computational properties, such as decidability and complexity, of several possible combinations are examined in detail in [32].

Quantitative aspects. Distance-based logics extend topological logics. Formulas are indexed by intervals, which are used as constraints. Metric-topological properties are verified by a model if the topological part of the formula is verified, and the constraints are satisfied. For example, one may require that points satisfying a certain property are located at most at a specified distance from each other, or from points characterised by some other property.

This tutorial does not cover several topics that are relevant for spatial and spatio-temporal logics and model checking. A partial list of well-known approaches that are not covered in this chapter includes: *Metric Interval Temporal Logics* [34]; *Spatial Signal Temporal Logic* [37,38]; *SpaTeL* [30] a spatial logic based on Quad Trees; logics of process calculi such as the Mobile Stochastic Logic *MoSL* [23], *Ambient Logic* [13] and *Separation Logic* [39]. Finally, we will only touch upon extensions with quantitative features in the final section. The spatial temporal logic based on Quad trees and spatial representations and non-logic based spatial analysis techniques will be addressed in other chapters of this volume [8,26].

The outline of the tutorial is as follows. In Section 2 we present and relate several classes of spatial structures that are directly relevant to the approach we follow to develop spatial logics and model-checking techniques. Section 3 reviews modal logics and their extension to space. In Section 4 a spatial logic for closure spaces is introduced and its operators are illustrated in a number of examples. In Section 6 the spatial logic is extended with temporal operators and in Section 7 some aspects of a spatio-temporal model-checking algorithm are presented. Section 8 presents some larger case studies in which we illustrate some more complicated spatio-temporal properties of a bike-sharing system and public bus transportation seen as a CAS. To conclude, Section 9 provides a discussion of open issues. The detailed proofs of theorems, lemmas and propositions stated in this tutorial can be found in [17].

2 Closure Spaces, Topology and Graphs

In this section we introduce some relevant mathematical notions and facts that are used throughout the tutorial and that form the basis for the particular spatial logics and related model checking algorithms that are the main topic of this tutorial. We briefly explain the various mathematical structures that are involved, in particular topological structures, their generalisation, also known as Closure Spaces, and two particular discrete subsets of the latter, namely *quasi-discrete closure spaces* and finite *graphs*. This section is intended as a minimal reference tailored to make the tutorial self-contained, rather than as an introduction to topology or other mathematical subjects, for which the reader is invited to consult more authoritative sources. The section on topology of [41] may be used as a gentle introduction; a comprehensive reference is [31].

2.1 Topological spaces

Topological spaces may be presented as generalisations of Euclidean spaces by focussing on the notion of *closeness* without making reference to an explicit metric. In topology this notion is expressed in terms of relationships between sets of points instead of in terms of distance. Any topological space consists of a set of points, a set of subsets of points called *open sets* and two operators on sets, namely union and intersection. More formally, a topological space can be defined as follows.

Definition 1. *A* topological space *is a pair (X, O) of a set X and a collection $O \subseteq \wp(X)$ of subsets of X called* open sets*, such that $\emptyset, X \in O$, and subject to closure under arbitrary unions and finite intersections.*

The dual of an open set is a *closed* set, but open and close sets are not mutually exclusive, as illustrated in Example 1 below following the definition of closed set.

Definition 2. *A subset S of X is called* closed *if $X \setminus S \in O$. A* clopen *is a set that is both open and closed.*

Example 1. Assume the topological space X is the two dimensional Euclidean space and that it is equipped with the usual Euclidean (metric) topology. The pink shapes in Figure 1 are subsets of this space. The shape on the left is open, that on the right is closed. An example of a set that is both open and closed is the space X itself. This can be seen as follows. First note that X is open by definition, but its complement, the empty set, is also open by definition. Therefore X is also closed. The same holds for the empty set. Furthermore, switching to the one dimensional case for simplicity, the set $[1, 2)$ is *neither* open *nor* closed because its complement is neither open nor closed. However, $[1, \infty)$ is a closed set because its complement is open.

Also the notion of *neighbourhood* of a point in space plays an important role, as well as those of *interior* and *closure* of a set of points.

Definition 3. *An* open neighbourhood *of $x \in X$ is an open set o with $x \in o$. The* interior *of $S \subseteq X$, denoted by $\mathcal{I}(S)$, is the largest open set contained in S and the* closure *of S, denoted by $\mathcal{C}(S)$, is the smallest closed set containing S.*

open set
closed set

Fig. 1: Example of an open set (left) and a closed set (right).

The interior and closure are dual. Let $\overline{S}$ denote $X \setminus S$ (the complement of S in X). Then we have $\mathcal{I}(S) = \overline{\mathcal{C}(\overline{S})}$ and $\mathcal{C}(S) = \overline{\mathcal{I}(\overline{S})}$. In Figure 1 the open set can be seen as the interior of the closed set and the closed set as the closure of the open set. In fact, the closure operator adds all *limit points* of an open set to it. A point p is called a *limit point* of a set S if every open set containing p also contains some point of S. Limit points are a fundamental notion in topological spaces and could be used to provide an alternative axiomatical definition of topological spaces known as Kuratowsky spaces. They also reflect the inherent *continuous* aspect of topological spaces. In topological spaces the closure operator is idempotent, so $\mathcal{C}(\mathcal{C}(S)) = \mathcal{C}(S)$.

2.2 Closure Spaces

Discrete spatial structures could be treated as in the continuous case, by defining a topology on top of the points of the structure. However, by doing so, one does not gain much, as the closure operator is idempotent in topological spaces. This assumption becomes too stringent for discrete structures. For example, in the case of regular grids, it is natural to interpret closure as the operation of enlarging a set of points by one step (in all possible directions) on the grid. Such interpretation is clearly not idempotent. By removing the idempotency assumption, *closure spaces* are obtained that therefore are a generalisation of topological spaces, as shown in the following definition and more explicitly in Definition 7.

Definition 4. *A* closure space *is a pair* $(X, \mathcal{C})$ *where* X *is a set, and* $\mathcal{C} : 2^X \to 2^X$ *assigns to each subset of* X *its* closure*, such that, for all* $A, B \subseteq X$*:*

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

As in topological spaces, we can define the interior operator for closure spaces as well as the notions of neighbourhood, open set and closed set.

Definition 5. *In a closure space* $(X, \mathcal{C})$*, given* $A \subseteq X$ *and* $x \in X$*: (i) the* interior $\mathcal{I}(A)$ *of* A *is the set* $\overline{\mathcal{C}(\overline{A})}$*; (ii)* A *is a* neighbourhood *of* $x \in X$ *if and only if* $x \in \mathcal{I}(A)$*; and (iii)* A *is* closed *if* $A = \mathcal{C}(A)$ *and it is* open *if* $A = \mathcal{I}(A)$*.*

The above defined operators enjoy the following useful properties.

Proposition 1. *In a closure space $(X, \mathcal{C})$, for all $A, B \subseteq X$, the following holds: (i) $\mathcal{I}(A) \subseteq A$; (ii) A is open if and only if $\overline{A}$ is closed; (iii) $A \subseteq B \implies \mathcal{C}(A) \subseteq \mathcal{C}(B)$ and $\mathcal{I}(A) \subseteq \mathcal{I}(B)$; and (iv) the open sets are closed under finite intersections, and arbitrary unions.*

Below, we provide a definition of the *boundary* of a set A which is entirely given in terms of closure and interior, and coincides with the definition of boundary in a topological space. Moreover, in discrete spaces (such as, grids) it sometimes makes sense to consider just the part of the boundary of a set A which lies entirely within, or outside, A itself. We also define these notions.

Definition 6. *In a closure space $(X, \mathcal{C})$, the* boundary *of $A \subseteq X$ is defined as $\mathcal{B}(A) = \mathcal{C}(A) \setminus \mathcal{I}(A)$. The* interior boundary *is $\mathcal{B}^-(A) = A \setminus \mathcal{I}(A)$, and the* closure boundary *is $\mathcal{B}^+(A) = \mathcal{C}(A) \setminus A$.*

Proposition 2. *The following equations hold in a closure space:*

$$\mathcal{B}(A) = \mathcal{B}^+(A) \cup \mathcal{B}^-(A) \tag{1}$$

$$\mathcal{B}^+(A) \cap \mathcal{B}^-(A) = \emptyset \tag{2}$$

$$\mathcal{B}(A) = \mathcal{B}(\overline{A}) \tag{3}$$

$$\mathcal{B}^+(A) = \mathcal{B}^-(\overline{A}) \tag{4}$$

$$\mathcal{B}^+(A) = \mathcal{B}(A) \cap \overline{A} \tag{5}$$

$$\mathcal{B}^-(A) = \mathcal{B}(A) \cap A \tag{6}$$

$$\mathcal{B}(A) = \mathcal{C}(A) \cap \mathcal{C}(\overline{A}) \tag{7}$$

The axioms defining a closure space are also part of the definition of a *Kuratowski closure space*, which is an alternative definition of a *topological space*. More precisely, the only missing axiom that makes a closure space Kuratowski is idempotence[4], that is $\mathcal{C}(\mathcal{C}(A) = \mathcal{C}(A)$.

Definition 7. *A* topological space *is a closure space where the closure operator is idempotent, that is, for all $A \subseteq X$, $\mathcal{C}(\mathcal{C}(A)) = \mathcal{C}(A)$.*

The correspondence between the Kuratowski definition (Definition 7) and the open sets definition (Definition 1) can be sketched as follows. To view a topological space defined in terms of open sets as a closure space, one defines $\mathcal{C}(A)$ as the smallest closed set containing A. For the converse, one uses the definition of an open set in a closure space, as given in Definition 5 (noting that closure is already assumed to be idempotent, by the Kuratowski definition).

[4] When recovering the definition of a topological space via open sets from the Kuratowski definition, it is noteworthy that the preservation of binary unions is sufficient to prove that *arbitrary* unions of *open* sets are open.

2.3 Graphs as Closure Spaces

Discrete spatial structures typically come in the form of a graph. A graph is described by its set of nodes X and its *connectedness* binary relation $R \subseteq X \times X$. A closure operator $\mathcal{C}_R$ can be derived from R as follows.

Definition 8. *Given a set X and a relation $R \subseteq X \times X$, define the closure operator $\mathcal{C}_R(A)$ as follows: $\mathcal{C}_R(A) = A \cup \{x \in X \mid \exists a \in A.(x, a) \in R\}$.*

Proposition 3. *The pair $(X, \mathcal{C}_R)$ is a closure space.*

Closure operators obtained by Definition 8 are not necessarily idempotent. This is intimately related to reflexivity and transitivity of R, as shown by Lemma 11 in [27], that we rephrase below.

Lemma 1. *The operator $\mathcal{C}_R$ is idempotent if and only if the reflexive closure $R^{=}$ of R is transitive.*

Note that, when R is transitive, so is $R^{=}$, thus $\mathcal{C}_R$ is idempotent. The reverse implication is not true, as one may have $(x, y) \in R$, $(y, x) \in R$, but $(x, x) \notin R$.

Finally, we recall that in [28], a discrete variant of the topological definition of the boundary of a set A is given, for the case where a closure operator is derived by Definition 8 from a reflexive and symmetric relation. Therein, in Lemma 5, it is proved that the definition coincides with the one we provided (see Definition 6). The latter is entirely given in terms of closure and interior, and coincides with the definition of boundary in a topological space. Therefore we preferred to adopt it for the general case of a closure space in this tutorial.

2.4 Quasi-discrete Closure Spaces

We now discuss interesting structures that do not necessarily have idempotent closure. See also Lemma 9 of [27] and the subsequent statements. We shall see that there is a very strong relation between the definition of a *quasi-discrete* space, given below, and graphs.

Definition 9. *A closure space is* quasi-discrete *if and only if one of the following equivalent conditions holds:*

i) each $x \in X$ has a minimal neighbourhood[5] *N_x;*
ii) for each $A \subseteq X$, $\mathcal{C}(A) = \bigcup_{a \in A} \mathcal{C}(\{a\})$.

The following is proved as Theorem 1 in [27].

Theorem 1. *A closure space $(X, \mathcal{C})$ is quasi-discrete if and only if there is a relation $R \subseteq X \times X$ such that $\mathcal{C} = \mathcal{C}_R$.*

[5] A *minimal neighbourhood* of x is a set that is a neighbourhood of x and is included in all other neighbourhoods of x.

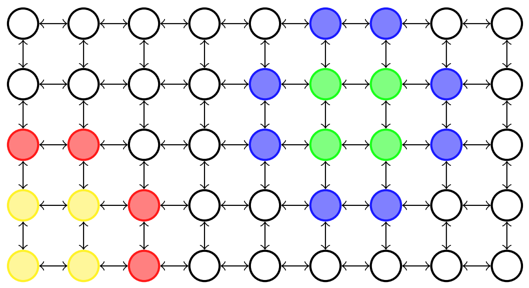


Fig. 2: A graph inducing a *quasi-discrete* closure space

Example 2. Existence of minimal neighbourhoods does not depend on finiteness of the space, and they do not even depend on the existence of a "closest element" for each point. To see this, consider the rational numbers $\mathbb{Q}$, equipped with the relation $\leq$. Such a relation is reflexive and transitive, thus the closure space $(\mathbb{Q}, \mathcal{C}_{\leq})$ is topological and quasi-discrete. •

Example 3. An example of a topological closure space which is not quasi-discrete is the set of real numbers equipped with the Euclidean topology (the topology induced by arbitrary union and finite intersection of open intervals). To see that the space is not quasi-discrete, one applies Definition 9. Consider an open interval (x, y). We have $\mathcal{C}((x, y)) = [x, y]$, but for each point z, we also have $\mathcal{C}(z) = [z, z] = \{z\}$. Therefore $\bigcup_{z \in (x,y)} \mathcal{C}(z) = \bigcup_{z \in (x,y)} \{z\} = (x, y) \neq [x, y]$. •

Example 4. The reader may think that quasi-discreteness is also related to the space having a smaller cardinality than that of the real numbers. This is not the case. To see this, just equip the real numbers with an arbitrary relation in a similar way to Example 2. The obtained closure space is quasi-discrete. •

Summing up, whenever one starts from an arbitrary relation $R \subseteq X \times X$, the obtained closure space $(X, \mathcal{C}_R)$ enjoys minimal neighbourhoods, and the closure of a set A is the union of the closure of the singletons composing A. Furthermore, such nice properties are only verified in a closure space when there is some R such that the closure operator of the space is derived from R. In the next example, we show some aspects of quasi-discreteness.

Example 5. Every graph induces a *quasi-discrete* closure space. For instance, we can consider the (undirected) graph depicted in Figure 2. Let R be the (symmetric) binary relation induced by the graph edges, and let Y and G denote the set of *yellow* and *green* nodes, respectively. The closure $\mathcal{C}_R(Y)$ consists of all *yellow* and *red* nodes, while the closure $\mathcal{C}_R(G)$ contains all *green* and *blue* nodes. The interior $\mathcal{I}(Y)$ of Y contains a single node, i.e. the one located at the bottom-left in Figure 2. On the contrary, the *interior* $\mathcal{I}(G)$ of G is empty. Finally, we have that $\mathcal{B}(G) = \mathcal{C}_R(G)$, while $\mathcal{B}^-(G) = G$ and $\mathcal{B}^+(G)$ consists of the *blue* nodes. •

2.5 Paths in Closure Spaces

A very useful notion to reason about spatial structures is that of paths. A uniform definition of paths for all closure spaces is, however, non-trivial. It is possible, and often done, to borrow the notion of path from topology. However, the extension is not fully satisfactory. For example, the topological definition does not yield graph-theoretical paths in the case of quasi-discrete closure spaces. As a pragmatic solution, we provide a natural definition of paths for interesting classes of closure spaces[6]. In particular, in this section we introduce the definition of continuous function, which restricts to topological continuity in the setting of idempotent closure spaces, and define paths in the case of quasi-discrete closure spaces. We postpone the discussion of paths for Euclidean topological space to Section 9.

Definition 10. *A* continuous function $f : (X_1, \mathcal{C}_1) \to (X_2, \mathcal{C}_2)$ *is a function* $f : X_1 \to X_2$ *such that, for all* $A \subseteq X_1$*, we have* $f(\mathcal{C}_1(A)) \subseteq \mathcal{C}_2(f(A))$.

Definition 11. *A (quasi-discrete) path for quasi-discrete closure space* $(X, \mathcal{C})$ *is a continuous function* $p : (\mathbb{N}, \mathcal{C}_{\mathrm{Succ}}) \to (X, \mathcal{C})$*, where* $(\mathbb{N}, \mathcal{C}_{\mathrm{Succ}})$ *is the closure space where* $(n, m) \in \mathrm{Succ} \iff m = n + 1$.

As a matter of notation, we call p a path *from* x, and write $p : x \rightsquigarrow \infty$, when $p(0) = x$. We write $y \in p$ whenever there is i such that $p(i) = y$.

It is worth noting that graph-theoretical and quasi-discrete paths coincide.

Proposition 4. *Given a (quasi-discrete) path* p *in a quasi-discrete space* $(X, \mathcal{C}_R)$*, for all* $i \in \mathbb{N}$ *with* $p(i) \neq p(i+1)$*, we have* $(p(i), p(i+1)) \in R$*, i.e., the image of* p *is a (graph theoretical, countably infinite) path in the graph of* R*. Conversely, each countable path in the graph of* R *uniquely determines a quasi-discrete path.*

2.6 Distance spaces and metric spaces

For the analysis of CAS often quantitative spatial information is required as well. Closure spaces can be enriched with such information by introducing *distance spaces* and *metric spaces*. We briefly mention some of them here. The interested reader may refer to Section 3.1 of [32] to get further insight on distance spaces. In particular, *qualitative* notions such as "being at a short distance" can be modelled in distance spaces but not in metric spaces.

Definition 12. *A* distance space *is a pair* (X, d) *of a set* X *and a function* $d : X \times X \to \mathbb{R}$ *such that, for all* $x, y \in X$, $d(x, y) = 0 \iff x = y$*, and* $d(x, y) \geq 0$*. If, in addition also* $d(x, y) = d(y, x)$ *and* $d(x, z) \leq d(x, y) + d(y, z)$ *hold, then the space is called a* metric space.

[6] We leave open the possibility to change this notion, in chosen classes of closure spaces, practically making our theory dependent on such choice. The theoretical question of finding a uniform notion of path is left for future work.

Definition 13. *A metric space can be equipped with the* metric topology *where the open sets are induced by the basis of* open balls, *that is, B is the collection of subsets $o \subseteq X$ such that there are $k \in \mathbb{R}$, $y \in X$ with $o = \{x \in X \mid d(x, y) < k\}$.*

The above definitions naturally extend to distance/metric closure spaces, when X is equipped with a closure operator $\mathcal{C}$.

2.7 Hierarchy of Closure Spaces

In Figure 3, the hierarchy of closure spaces with respect to quasi-discreteness is shown. All finite spaces are quasi-discrete, as closure of finite sets is determined by that of the singletons, by inductive application of the axiom $\mathcal{C}(A \cup B) = \mathcal{C}(A) \cup \mathcal{C}(B)$. Obviously, there are quasi-discrete infinite spaces (any infinite graph interpreted as a closure space is an example). A quasi-discrete space which is also topological is the space associated with any complete graph. In this case, for any set, $\mathcal{C}(A)$ is the whole space, thus closure is idempotent. One may wonder what kind of topology is associated with any complete graph, seen as a quasi-discrete closure space. This is precisely the *indiscrete* topology, which is also characterised, using the open sets definition, as the topology generated by a basis in which the only open sets are the empty set and the whole space. Indeed, there is another way to equip a set of points with a relation, in such a way that the resulting graph is an idempotent closure space. Namely, one can just consider the identity relation. Then, the closure operator of the obtained space is the identity function (which is clearly idempotent). Using the open sets definition, the obtained topology is the *discrete* topology, where the open sets are all the singletons. Finally there are closure spaces that are neither topological nor quasi discrete. The most obvious example is the *coproduct* (disjoint union) of a topological and a quasi-discrete, but not topological, closure space, which is a closure space under Definition 14.

Definition 14. *Given two closure spaces $(X, \mathcal{C}^X)$ and $(Y, \mathcal{C}^Y)$, consider the disjoint union of X and Y, represented as $X \uplus Y = X' \cup Y'$ with $X' = \{(1, x) \mid x \in X\}$ and $Y' = \{(2, y) \mid y \in Y\}$. In order to equip the set $X \uplus Y$ with a closure operator, for each $A \subseteq X \uplus Y$, let $A^X = \{x \mid (1, x) \in A\}$ and $A^Y = \{y \mid (2, y) \in A\}$. Define $\mathcal{C}(A) = \{(1, x) \mid x \in \mathcal{C}^X(A^X)\} \cup \{(2, y) \mid y \in \mathcal{C}^Y(A^Y)\}$.*

3 Modal Logics

Now that we have seen the various mathematical structures to represent space, we turn to *reasoning* about space and spatial properties. A successful way to reason about properties of structures in general is by way of a logic with which to express the properties of interest. Spatial logics have been mainly studied from the point of view of *modal* logics and there is a historical reason for that. In a seminal work of 1938, Tarski presented a spatial, and in particular topological, interpretation of modal logic, which paved the way for a new line of research

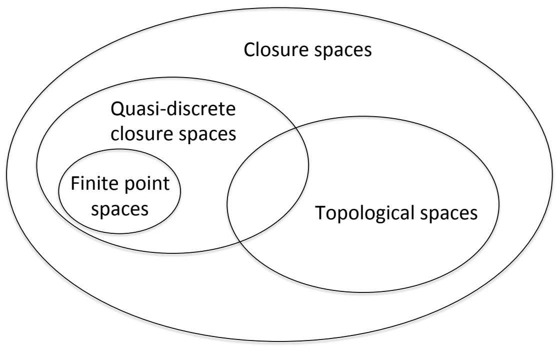


Fig. 3: The hierarchy of closure spaces.

on the relationship between topological spaces and modal logics, culminating in the proof, by Tarsky and McKinsey in 1944, that the simple (and decidable) modal logic $\mathcal{S}4$ is complete when interpreting the *possibility* modality $\Diamond$ of $\mathcal{S}4$ as *closure* on the reals or any similar metric space. The modal logics approach to spatial logic contemplates *purely spatial* logics, meaning that they deal with the spatial configuration of a system at a certain point in time, considering a particular 'snapshot', with no subsequent temporal evolution.

In this section we briefly recall the notions of modal logics that are directly relevant to the topic of this tutorial. For a more extended introduction to modal logics the reader is invited to consult more authoritative sources. A recommended reference for modal logics is [9], whereas for the study of their spatial intepretation we refer to [41].

3.1 Modal logics

We introduce the syntax of a basic modal logic, that we denote with $\mathcal{L}$, which forms the basis for most other logics presented in this tutorial.

Definition 15. *Fix a set of* proposition letters AP*, also called* Atomic Propositions*. Let p denote an arbitrary letter. The syntax of $\mathcal{L}$ is described by the grammar:*

$$\Phi ::= p \mid \top \mid \bot \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi \mid \Box\Phi \mid \Diamond\Phi$$

The relational semantics of $\mathcal{L}$ is given using *frames* and *models*.

Definition 16. *Fixed a set AP of* atomic propositions*, a* (Kripke) frame *is a pair (X, R) of a set X and an* accessibility relation $R \subseteq X \times X$*. A* model $\mathcal{M} = ((X, R), \mathcal{V})$ *consists of a frame (X, R) and a valuation $\mathcal{V} : AP \to \wp(X)$, assigning to each atomic proposition the set of points (also called 'possible worlds') that satisfy it.*

Truth of a formula is defined at a specific point $x \in X$.

Definition 17. *Truth* $\models$ *of modal formulas in model* $\mathcal{M} = ((X, R), \mathcal{V})$ *at point* $x \in X$ *is defined by induction as follows:*

$$\begin{array}{lll}
\mathcal{M}, x \models \top & \iff & \textit{true} \\
\mathcal{M}, x \models \bot & \iff & \textit{false} \\
\mathcal{M}, x \models p & \iff & x \in \mathcal{V}(p) \\
\mathcal{M}, x \models \neg\varphi & \iff & \texttt{not}\, \mathcal{M}, x \models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \iff & \mathcal{M}, x \models \varphi \texttt{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \Box\varphi & \iff & \forall y \in X.(x, y) \in R \implies \mathcal{M}, y \models \varphi \\
\mathcal{M}, x \models \Diamond\varphi & \iff & \exists y \in X.(x, y) \in R \wedge \mathcal{M}, y \models \varphi
\end{array}$$

The operators $\neg$, $\wedge$ and $\vee$ are the common Boolean operators negation, conjunction and disjunction, respectively. There are two modal operators. The operator $\Box\varphi$ denotes necessity and $\Diamond\varphi$ possibility. A point $x \in X$ satisfies *necessarily* φ, denoted as $\Box\varphi$, if φ holds in all points (worlds) y that are accessible from x via the accessibility relation R. A point $x \in X$ satisfies *possibly* φ, denoted as $\Diamond\varphi$, if there exists a point (world) y, accessible from x via the accessibility relation R, such that φ holds in y.

3.2 Modal logics of space

The presentation in this section is dealing with modal logics of space and is mostly based on the book chapter [41]. Many variants of spatial modal logics have been proposed. For instance, in *local topological logics* modalities identify *open sets* in which some or all points ought to satisfy a given property, while in *global topological logics* it is possible, in addition, to predicate about the satisfaction of a certain property by classes of points in the space (e.g., *all* points) and in *distance logics* truth depends upon some notion of distance between entities, just to mention a few. In the following we only address the local topological variant in some more detail and illustrate the limitations of a purely topological approach when we are interested in a spatial logic for the wider class of spatial structures that are of interest for CAS, which includes discrete spatial structures.

Topo-models and topo-logics Following the approach of Tarski, a topological space (Definition 1) may be used in place of a *frame* (see Definition 16) in order to interpret the modal logic $\mathcal{L}$ (Definition 15), obtaining *topological modal logics* or simply *topo-logics*. We first need to define a topological *model*.

Definition 18. *Fixed a set AP of* Atomic Propositions*, a* topological model *or* topo-model $\mathcal{M} = ((X, O), \mathcal{V})$ *consists of a topological space* (X, O) *and a valuation* $\mathcal{V} : AP \to \wp(X)$*, assigning to each atomic proposition the set of points that it satisfies.*

Truth of a formula is defined at a specific point x in space X.

Definition 19. *Truth $\models$ of modal formulas in model $\mathcal{M} = ((X, O), \mathcal{V})$ at point $x \in X$ is defined by induction as follows:*

$$
\begin{array}{lcl}
\mathcal{M}, x \models \top & \iff & \textit{true} \\
\mathcal{M}, x \models \bot & \iff & \textit{false} \\
\mathcal{M}, x \models p & \iff & x \in \mathcal{V}(p) \\
\mathcal{M}, x \models \neg\varphi & \iff & \texttt{not}\, \mathcal{M}, x \models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \iff & \mathcal{M}, x \models \varphi \texttt{ and } \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \Box\varphi & \iff & \exists o \in O.(x \in o \texttt{ and } \forall y \in o.\mathcal{M}, y \models \varphi) \\
\mathcal{M}, x \models \Diamond\varphi & \iff & \forall o \in O.(x \in o \texttt{ implies } \exists y \in o.\mathcal{M}, y \models \varphi)
\end{array}
$$

The usual De Morgan-style dualities hold, including $\mathcal{M}, x \models \Box\varphi \iff \mathcal{M}, x \models \neg\Diamond\neg\varphi$. The interpretation of formulas identifies regions of space that depend on the valuation $\mathcal{V}$. In particular, note that the operation $\Box\varphi$ identifies the topological *interior* of the region where φ holds. Dually, $\Diamond\varphi$ denotes the topological *closure* of φ. An example formula which is widely used is the *boundary* of a property, which can be introduced as a derived operator: $\mathcal{B}\varphi \triangleq \Diamond\varphi \wedge \neg\Box\varphi$.

Example 6. We report in Figure 4 the first example from [41]. The topological space here is the two-dimensional Euclidean plane $\mathbb{R}^2$ equipped with the metric topology. The only proposition letter is p and the valuation of p assigns to this property the shape of a "spoon" composed of a line segment and a filled ellipse. Various formulas can denote regions such as the boundary of the spoon, including or excluding the handle, the inner part of the spoon, the whole figure without the handle, etc. The boundary and the handle are drawn much thicker than they really are only to show them more clearly.

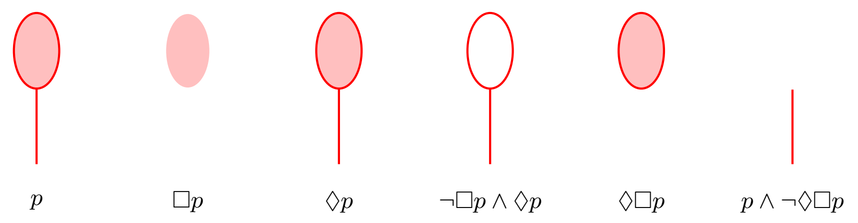


Fig. 4: Topological interpretation of formulas over a topo-model. From left to right: all points satisfying atomic proposition p; the interior of the region where p holds; the closure of the region where p holds; the closure without the interior points satisfying p; the closure of the interior of the points satisfying p; those points satisfying p that are not in the closure of the interior.

Axiomatic aspects and relational semantics From the point of view of logics, it is important to understand the axioms and the deductive power of a logic, and in particular its completeness with respect to classes of models. A logic is complete with respect to a class of models C, if all formulas that are true in every model in C, i.e. they are *valid* in C, are also provable using the axioms and rules of the logic. For such a statement to make sense in the setting of topo-logics, one needs to specify that a formula φ is true in a model $\mathcal{M} = ((X, O), \mathcal{V})$ if $\mathcal{M}, x \models \varphi$ for all $x \in X$. Once this is established, various axioms are considered. As an example, we show those of the logic $\mathcal{S}4$, together with the relevant theorem. The logic $\mathcal{S}4$ is the modal logic $\mathcal{L}$ obtained when the accessibility relation of the frame is transitive and reflexive. We refer the reader to [41] for further details.

Definition 20. *The logic $\mathcal{S}4$ is $\mathcal{L}$ under the axioms K, T, 4.*

$$\begin{array}{ll} \Box(p \to q) \to (\Box p \to \Box q) & \textit{(K) distributivity} \\ \Box p \to \Box\Box p & \textit{(4) transitivity} \\ \Box p \to p & \textit{(T) reflexivity} \end{array}$$

These axioms further clarify the properties of topo-logics, as follows.

Theorem 2. *Assume the rules for* modus ponens *and* necessitation*:*

$$\frac{\varphi \quad \varphi \to \psi}{\psi} \qquad \frac{\varphi}{\Box\varphi}$$

The logic $\mathcal{S}4$ is complete with respect to topological models, that is, whenever φ is valid, it can be proved using the axioms $K, 4, T$, using modus ponens and necessitation.

Having seen this, and knowing that there are relational models of $\mathcal{S}4$, that is, the reflexive and transitive Kripke frames, one may wonder whether the connection is deeper. This is analysed in Section 2.4.1 of [41]. It is possible to derive a topological space from a frame, and the other way around, in a sound and complete way. The topological spaces that are used are the so-called *Alexandroff spaces*. These are spaces in which each point has a least open neighbourhood.

The correspondence between topological spaces and reflexive and transitive Kripke frames is not easily extended to arbitrary frames, as transitivity and reflexivity always hold in topo-logics where the basic modality is the closure.

On the other hand, requiring transitivity in all models may be a too limiting constraint, e.g., when "one-way" links are to be taken into account. This is the main reason to further investigate non-transitive concepts of spatial models in the context of closure spaces, and in particular quasi-discrete ones.

4 Spatial Logic for Quasi-discrete Closure Spaces

Whereas the local variant of modal logics of space follows a purely topological approach, in this section we lift the topological definitions to the more general setting of closure spaces and extend this framework with further spatial operators, in particular a spatial surrounded operator. The resulting logic is SLCS: a

Spatial Logic for Closure Spaces, that we first proposed in [18]. In this section we focus our attention to quasi-discrete closure spaces; some ideas on how to generalize definitions and results to a broader class of spaces, including for instance the Euclidean topological space, will be discussed in Section 9.

SLCS is meant to assign to formulas a *local* meaning; for each point, formulas may predicate both on the possibility of reaching other points satisfying specific properties, or of being reached from them, along paths of the space. In [18], SLCS is equipped with two *spatial operators*: a "one step" modality, called "near" and denoted by $\mathcal{N}$, turning the closure operator $\mathcal{C}$ into a logical operator, and a binary *spatial until* operator $\mathcal{U}$, which is a spatial counterpart of the temporal *until* operator that is part of many well-known temporal logics such as CTL (for a description see e.g. [4]). In this tutorial we denote this last operator by $\mathcal{S}$ in order to avoid confusion in later sections where SLCS is further combined with temporal operators such as the temporal until operator $\mathcal{U}$ leading to a spatio-temporal logic.

Assume a finite or countable set AP of *atomic propositions*. The syntax of SLCS is defined by the grammar in Figure 5, where p ranges over AP. In

$$\begin{array}{lll}
\Phi ::= & p & [\textsc{Atomic proposition}] \\
\mid & \top & [\textsc{True}] \\
\mid & \neg\Phi & [\textsc{Not}] \\
\mid & \Phi \wedge \Phi & [\textsc{And}] \\
\mid & \mathcal{N}\Phi & [\textsc{Near}] \\
\mid & \Phi\,\mathcal{S}\,\Phi & [\textsc{Surrounded}]
\end{array}$$

Fig. 5: SLCS syntax

Figure 5, $\top$ denotes the truth value *true*, $\neg$ is negation, $\wedge$ is conjunction, $\mathcal{N}$ is the *closure* operator, $\mathcal{S}$ is the *surrounded* operator. From now on, with a small overload of notation, we let Φ denote the set of SLCS formulas. Next we define the interpretation of formulas.

Definition 21. *A* closure model *is a pair $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$ consisting of a closure space $(X, \mathcal{C})$ and a valuation $\mathcal{V} : AP \to 2^X$, assigning to each atomic proposition the set of points where it holds. Whenever $(X, \mathcal{C})$ is quasi-discrete, $\mathcal{M}$ is called a* quasi-discrete closure model*.*

Definition 22. *Satisfaction $\mathcal{M}, x \models \varphi$ of formula φ at point x in quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$ is defined, by induction on terms, as follows:*

$$\begin{array}{lcl}
\mathcal{M}, x \models p & \iff & x \in \mathcal{V}(p) \\
\mathcal{M}, x \models \top & \iff & \mathit{true} \\
\mathcal{M}, x \models \neg\varphi & \iff & \texttt{not}\ \mathcal{M}, x \models \varphi \\
\mathcal{M}, x \models \varphi \wedge \psi & \iff & \mathcal{M}, x \models \varphi\ \texttt{and}\ \mathcal{M}, x \models \psi \\
\mathcal{M}, x \models \mathcal{N}\varphi & \iff & x \in \mathcal{C}(\{y \in X | \mathcal{M}, y \models \varphi\}) \\
\mathcal{M}, x \models \varphi\,\mathcal{S}\,\psi & \iff & \exists A \subseteq X. x \in A \wedge \forall y \in A. \mathcal{M}, y \models \varphi \wedge \\
 & & \quad \forall z \in \mathcal{B}^{+}(A). \mathcal{M}, z \models \psi
\end{array}$$

Atomic propositions and boolean connectives have the expected meaning. For formulas of the form $\varphi_1 \, \mathcal{S} \, \varphi_2$, the basic idea is that point x satisfies $\varphi_1 \, \mathcal{S} \, \varphi_2$ whenever there is "no way out" from a set of points, including x, and that each satisfy φ_1 unless passing by a point that satisfies φ_2. For instance, if we consider the model of Figure 2, *yellow* nodes should satisfy $\mathit{yellow} \, \mathcal{S} \, \mathit{red}$ while *green* nodes should satisfy $\mathit{green} \, \mathcal{S} \, \mathit{blue}$.

In Figure 6, we present some derived operators. Besides standard logical connectives, the logic can express the *interior* ($\mathcal{I}\varphi$), the *boundary* ($\delta\varphi$), the *interior boundary* ($\delta^{-}\varphi$) and the *closure boundary* ($\delta^{+}\varphi$) of the set of points satisfying formula φ. Moreover, by appropriately using the *surrounded* operator, operators concerning *reachability* ($\varphi_1 \, \mathcal{R} \, \varphi_2$), *from-to* ($\varphi_1 \, \mathcal{T} \, \varphi_2$), *global satisfaction* ($\mathcal{E} \, \varphi$, *everywhere* φ) and *possible satisfaction* ($\mathcal{F}\varphi$, *somewhere* φ) can be derived.

A point x satisfies $\varphi_1 \, \mathcal{R} \, \varphi_2$ 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 $(\mathit{white} \vee \mathit{blue}) \, \mathcal{R} \, \mathit{blue}$, 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 $\mathcal{T}$. The operator $\mathcal{T}$ is a slightly stronger version of the reachability operator $\mathcal{R}$ requiring that there is a path from a point satisfying φ_1, reaching a point satisfying φ_2 while passing only by points satisfying φ_1. Note that φ_2 is occurring also in the first argument of $\mathcal{R}$ in the definition of $\mathcal{T}$. This is because satisfaction of $\varphi_1 \, \mathcal{R} \, \varphi_2$ requires that the final node on the path satisfies both φ_1 and φ_2. We show further examples of the use of $\mathcal{T}$ shortly.

An interesting observation is that the modal spatial operators can also be characterised by definitions based on quasi-discrete paths.

Proposition 5. *We have that:*

1. $\mathcal{M}, x \models \mathcal{N}\varphi$ *if and only if there is* y *and* $p : y \rightsquigarrow \infty$ *such that* $p(0) = y$ *and* $\mathcal{M}, y \models \varphi$ *and* $p(1) = x$*;*
2. $\mathcal{M}, x \models \varphi_1 \, \mathcal{S} \, \varphi_2$ *if and only if* $\mathcal{M}, x \models \varphi_1$ *and for all* $p : x \rightsquigarrow \infty . \forall l . \mathcal{M}, p(l) \models \neg\varphi_1$ *implies* $\exists k . 0 < k \leq l . \mathcal{M}, p(k) \models \varphi_2$*;*
3. $\mathcal{M}, x \models \varphi_1 \, \mathcal{R} \, \varphi_2$ *if and only if there is* $p : x \rightsquigarrow \infty$ *and* k *such that* $\mathcal{M}, p(k) \models \varphi_2$ *and for each* j *with* $0 < j \leq k$*, we have* $\mathcal{M}, p(j) \models \varphi_1$*;*
4. $\mathcal{M}, x \models \mathcal{E} \, \varphi_1$ *if and only if for each* $p : x \rightsquigarrow \infty$ *and* $i \in \mathbb{N}$*,* $\mathcal{M}, p(i) \models \varphi_1$*;*
5. $\mathcal{M}, x \models \mathcal{F}\varphi_1$ *if and only if there is* $p : x \rightsquigarrow \infty$ *and* $i \in \mathbb{N}$ *such that* $\mathcal{M}, p(i) \models \varphi_1$.

We conclude this section by pointing out that digital images are a noteworthy example of quasi-discrete closure models.

$$\begin{array}{llll}
\bot & \triangleq \neg\top & \varphi_1 \vee \varphi_2 & \triangleq \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\mathcal{I}\varphi & \triangleq \neg(\mathcal{N}\neg\varphi) & \delta\varphi & \triangleq (\mathcal{N}\varphi) \wedge (\neg\mathcal{I}\varphi) \\
\delta^{-}\varphi & \triangleq \varphi \wedge (\neg\mathcal{I}\varphi) & \delta^{+}\varphi & \triangleq (\mathcal{N}\varphi) \wedge (\neg\varphi) \\
\varphi_1 \,\mathcal{R}\, \varphi_2 & \triangleq \neg((\neg\varphi_2)\,\mathcal{S}(\neg\varphi_1)) & \mathcal{E}\,\varphi & \triangleq \varphi\,\mathcal{S}\,\bot \\
\varphi_1 \,\mathcal{T}\, \varphi_2 & \triangleq \varphi_1 \wedge ((\varphi_1 \vee \varphi_2)\,\mathcal{R}\,\varphi_2) & \mathcal{F}\varphi & \triangleq \neg(\mathcal{E}\,\neg\varphi)
\end{array}$$

Fig. 6: Some SLCS derived operators

Example 7. Any digital image can be treated as a finite, thus quasi-discrete model. Consider the closure space consisting of the plane $X = \mathbb{N} \times \mathbb{N}$, equipped with the closure operator $\mathcal{C}_{4adj}$ defined by the 4-adjacency relation of digital topology, namely:

$$((x_1, y_1), (x_2, y_2)) \in 4adj \iff ((x_1 - x_2) + (y_1 - y_2))^2 = 1$$

In words, such closure space is a regular grid where each pixel, except those on the borders, has four neighbours, corresponding to the directions right, left, up and down[7]. On top of this space, atomic propositions can be interpreted as specifications of colours (such as, RGB coordinates, ranges, etc.), so that each point (x, y) satisfies precisely those specifications that include the colour of the pixel at coordinates (x, y). •

5 Spatial Model Checking

In [18] algorithms for spatial model checking of SLCS are presented. They are available as a proof-of-concept tool[8] called `topochecker`. The tool is implemented in OCaml[9], 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 more interesting part of the algorithm is that for the surrounded operator, shown in Figure 7. Function `Sat`, computed by Algorithm 1, implements the model checker for SLCS. The function takes as input a finite, quasi-discrete model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$ and a SLCS formula φ, and returns the set of all points in X satisfying φ. The function is inductively defined on the structure of φ and, following a bottom-up approach, computes the resulting set via an appropriate combination of the recursive invocations of `Sat` on the subformulas of φ. When φ is of the form $\top$, p, $\neg\varphi_1$ or $\varphi_1 \wedge \varphi_2$, the definition of $\texttt{Sat}(\mathcal{M}, \varphi)$ is straightforward. To compute the set of points satisfying $\mathcal{N}\varphi_1$, the closure operator $\mathcal{C}$ of the space is applied to the set of points satisfying φ_1. When φ is of the form $\varphi_1\,\mathcal{S}\,\varphi_2$, function `Sat` relies on the function `CheckSurr` defined in Algorithm 2.

[7] This notion of neighbourhood is also known as the von Neumann neighbourhood of radius 1.

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

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

This global flooding algorithm and its operation is illustrated in an informal way in Figure 8 for the formula *yellow* $\mathcal{S}$ *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.

```
Function Sat($\mathcal{M}, \varphi$)
    Input: Finite, quasi-discrete
           closure model
           $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$, formula $\varphi$
    Output: Set of points
           $\{x \in X \mid \mathcal{M}, x \models \varphi\}$
    Match $\varphi$
        case $\top$ : return $X$
        case $p$ : return $\mathcal{V}(p)$
        case $\neg\varphi_1$ :
            let $P = \mathtt{Sat}(\mathcal{M}, \varphi_1)$
            return $X \setminus P$
        case $\varphi_1 \wedge \varphi_2$ :
            let $P = \mathtt{Sat}(\mathcal{M}, \varphi_1)$
            let $Q = \mathtt{Sat}(\mathcal{M}, \varphi_2)$
            return $P \cap Q$
        case $\mathcal{N}\varphi_1$ :
            let $P = \mathtt{Sat}(\mathcal{M}, \varphi_1)$
            return $\mathcal{C}(P)$
        case $\varphi_1 \mathcal{S} \varphi_2$ :
            return CheckSurr
            ($\mathcal{M}$,$\varphi_1$,$\varphi_2$)
```

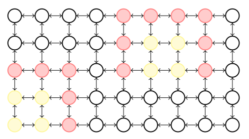
Algorithm 1: Decision procedure for the model checking problem of SLCS.

```
Function CheckSurr ($\mathcal{M}$,$\varphi_1$,$\varphi_2$)
    Input: Finite, quasi-discrete
           closure model
           $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$,
           formulas $\varphi_1, \varphi_2$
    Output: Set of points $\{x \in X \mid$
           $\mathcal{M}, x \models \varphi_1 \mathcal{S} \varphi_2\}$
    var $V := \mathtt{Sat}(\mathcal{M}, \varphi_1)$
    let $Q = \mathtt{Sat}(\mathcal{M}, \varphi_2)$
    var $T := \mathcal{B}^+(V \cup Q)$
    while $T \neq \emptyset$ do
        var $T' := \emptyset$
        for $x \in T$ do
            let $N = pre(x) \cap V$
            $V := V \setminus N$
            $T' := T' \cup (N \setminus Q)$
        $T := T'$;
    return $V$
```

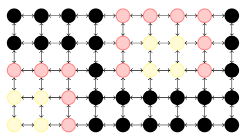
Algorithm 2: Checking *surrounded* formulas in a quasi-discrete closure space.

Fig. 7: Spatial model checking algorithm: surrounded operator.

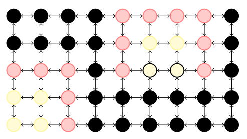
As an illustration, we show the application of the spatial model checker on an image (see Figure 9) representing a maze, in which the pixels form the points of a finite quasi-discrete closure space and in which the relation between points is given as a regular graph, much like in Figure 2, but not shown in the image. The green area is the exit. The blue areas (rectangular and round spots) are starting points.



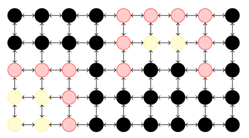
a) Given space



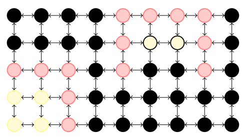
b) Identify all states not red or yellow



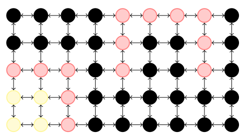
c) Select yellow states neighbours of black ones



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

Fig. 8: Flooding algorithm evaluating *Yellow* $\mathcal{S}$ *Red*

All model checking examples in this tutorial use `topochecker` as supporting tool. Consequently, we use the tool's frontend syntax for the SLCS formulas in such examples. The correspondence is straightforward.

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 $\mathcal{S}$ operator, by means of the derived operators $\mathcal{R}$ and $\mathcal{T}$, discussed earlier. 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.

$$\texttt{toExit} = [\texttt{white}]\, \texttt{T}\, [\texttt{green}]$$

The model checking result is shown in Figure 10. Points that satisfy this formula are the yellow and orange ones[10].

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).

$$\texttt{fromStartToExit} = \texttt{toExit}\,\&\,([\texttt{white}]\, \texttt{T}\, [\texttt{blue}])$$

Points that satisfy this formula are the orange ones in Figure 10.

[10] Actually one colour (yellow) could have been used, but in order to show multiple verification results combined in one picture, the orange points show the points that are yellow but that also satisfy the second property.

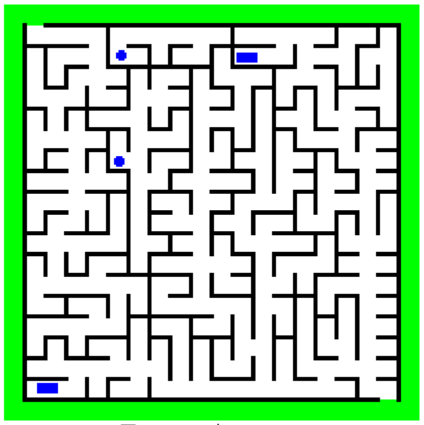


Fig. 9: A maze.

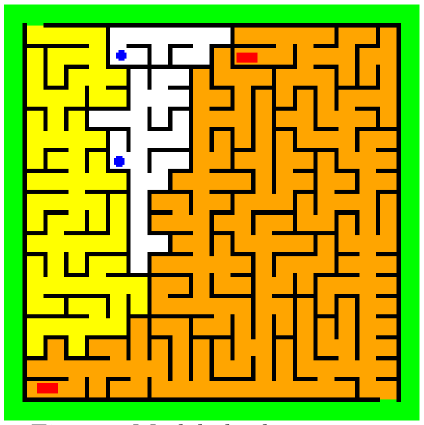


Fig. 10: Model checker output.

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.

$$\texttt{startCanExit} = [\texttt{blue}]\, \mathcal{T}\, \texttt{fromStartToExit}$$

Points that satisfy this formula are shown in red in Figure 10. These are indeed only the two small 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.

A further example is the use of spatial model checking to study the formation of patterns in bio-chemical systems. Alan Turing conjectured in [40] that pattern formation is a consequence of the coupling of reaction and diffusion phenomena involving different chemical substances distributed in the same physical space. Such behaviour can be described by a set of reaction-diffusion equations. We show here a variant in which wave-like patterns emerge. This is a variant of the models used in [3,30,38].

We use a reaction-diffusion system that is discretised, according to a Finite Difference scheme, as a system of ordinary differential equations whose variables are organised in a $K \times K$ rectangular grid. As before, such a grid is considered as a bi-directional graph, where each node $(i,j) \in L = \{1,\ldots,K\} \times \{1,\ldots,K\}$ represents a discrete location, edges connect pairs of neighbouring nodes along four directions as in the 4-adjacency relation in digital topology of Example 7.

We consider two chemical substances A and B in the $K \times K$ grid, obtaining the system:

$$\begin{cases} \frac{dx^A_{i,j}}{dt} = R_1 x^A_{i,j} x^B_{i,j} - x^A_{i,j} + R_2 + D_1(\mu^A_{i,j} - x^A_{i,j}) & i = 1..,K,\ j = 1,..,K, \\ \frac{dx^B_{i,j}}{dt} = R_3 x^A_{i,j} x^B_{i,j} + R_4 + D_2(\mu^B_{i,j} - x^B_{i,j}) & i = 1..,K,\ j = 1,..,K, \end{cases} \tag{8}$$

where: $x_{i,j}^A$ and $x_{i,j}^B$ are the concentrations of the two chemical substances in the location (i,j); R_i, $i = 1, ..., 4$ are the parameters that define the reaction between the two species; D_1 and D_2 are the diffusion constants, i.e. constants that define the movement of the molecules between locations; $\mu_{i,j}^A$ and $\mu_{i,j}^B$ are the average concentrations of the locations adjacent to location (i,j), that is

$$\mu_{i,j}^n = \frac{1}{|\nu_{i,j}|} \sum_{\nu \in \nu_{i,j}} x_\nu^n \qquad n \in \{A, B\}, \tag{9}$$

where $\nu_{i,j}$ is the set of indices of locations adjacent to (i,j). Note that if the average concentration of a substance in the adjacent nodes of (i,j) is higher than in the node (i,j) itself, then there is a flow of the substance entering (i,j) (assuming that the diffusion parameter has a positive value). If the average concentration is lower, there is a flow going out of (i,j) to its neighbours.

The flow towards a location could happen in two ways. In the first case, all neighbours are equally "attractive", and so the location (i,j) could receive an equal proportion from each of the adjacent locations. This leads to the formation of patterns with more or less round "spots" as shown also in [3,30,38]. However, we can also study cases in which there is a preferred direction of such flow, for example the location receives relatively more substance from the neighbours located north-west of it than from those located south-east of it. This is just one of the possibilities and it leads to the formation of rather different patterns. We can introduce such a bias by multiplying the concentrations of the four neighbours by different weights, k_i with $i \in \{\text{north}, \text{south}, \text{west}, \text{east}\}$ in such a way that their sum is equal to 4 (as would have been the case when they would all have the same weight).

$$\mu_{i,j}^n = \frac{1}{|\nu_{i,j}|} (k_{\text{north}} x_{\nu_{i,j+1}}^n + k_{\text{south}} x_{\nu_{i,j-1}}^n + k_{\text{west}} x_{\nu_{i-1,j}}^n + k_{\text{east}} x_{\nu_{i+1,j}}^n) \tag{10}$$

An example of the evolution of the concentration of substance A is shown in Figure 11 for snapshots of the space at various points in time. These snapshots are obtained from a numerical solution of Eq. (8) by standard tools such as Octave[11]. The following values were used for the parameters: $K = 32$, $R1 = 1$, $R2 = -12$, $R3 = -1$, $R4 = 16$, $D1 = 5.6$ and $D2 = 25.5$. Whereas $k_{\text{north}} = 1.5$, $k_{\text{south}} = 0.5$, $k_{\text{west}} = 1.5$ and $k_{\text{east}} = 0.5$. The initial condition is set randomly.

Using spatial model checking we can easily identify the points in which the concentration a of A is smaller than 2 and that are surrounded by points in which the concentration is higher than 2.

$$\texttt{pattern} = [a < 2]\texttt{S}\ [a > 2]$$

The points that satisfy property `pattern` in the snapshot at $t = 10$ are shown in Figure 12 (left). In this figure neither the edges of the graph nor the nodes where the formula is not satisfied are shown to avoid cluttering the image.

[11] See `http://octave.sourceforge.net/`

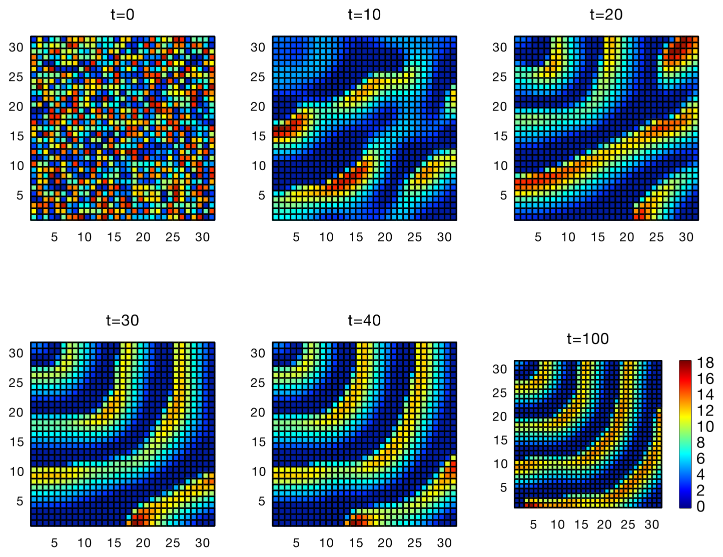


Fig. 11: Evolution of a wave-like pattern at different points in time. The colours indicate the concentration of substance A.

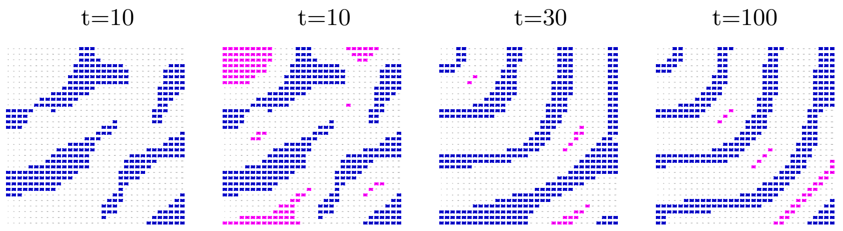


Fig. 12: Points that satisfy `pattern` in the snapshot at t=10 (left); points that satisfy property `far_from_pattern` are shown in pink in the other three snapshots at $t = 10$, $t = 30$ and $t = 100$, respectively; blue points satisfy property `pattern`.

The following formula provides an indication of the regularity of the pattern. It is satisfied by points that are at least 4 steps away from the areas with low concentration. These are shown in pink in Figure 12 (centre and right) for $t = 10$, $t = 30$ and $t = 100$. The points in blue are the ones satisfying property `pattern`. For more regular patterns fewer points are satisfying the property. If we would have verified the property for a 'distance' of 5 instead of 4 then in the snapshot of $t = 100$ none of the points would satisfy the property. This can been checked using the somewhere operator $\mathcal{F}$ by verifying $\neg\mathcal{F}$`far_from_pattern` (not shown in the figure).

$$\texttt{far\_from\_pattern} = !(\texttt{N}\,(\texttt{N}\,(\texttt{N}\,(\texttt{N}\,(\texttt{pattern})))))$$

6 Spatio-temporal Logic of Closure Spaces

Starting from a spatial formalism and 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. For spatial logics based on *topological* spaces a large number of possibilities are explored in [32]. We investigated one such structure, in the setting of closure spaces, namely the combination of the temporal logic *Computation Tree Logic* (CTL) (see e.g. [4]) and of SLCS, resulting in the *Spatio-Temporal Logic of Closure Spaces* (STLCS). In STLCS spatial and temporal fragments may be arbitrarily and mutually nested.

First, we show the formal syntax of formulas, described by the grammar in Figure 13, where p ranges over a finite or countable set of atomic propositions AP.

$$\begin{array}{rll@{\qquad}rll}
\Phi ::= & \top & [\textsc{True}] & \varphi ::= & \mathcal{X}\,\Phi & [\textsc{Next}] \\
\mid & p & [\textsc{Atomic predicate}] & \mid & \mathtt{F}\,\Phi & [\textsc{Eventually}] \\
\mid & \neg\,\Phi & [\textsc{Not}] & \mid & \mathtt{G}\,\Phi & [\textsc{Globally}] \\
\mid & \Phi \wedge \Phi & [\textsc{And}] & \mid & \Phi\,\mathcal{U}\,\Phi & [\textsc{Until}] \\
\mid & \mathcal{N}\,\Phi & [\textsc{Near}] & & & \\
\mid & \Phi\ \mathcal{S}\ \Phi & [\textsc{Surrounded}] & & & \\
\mid & \mathtt{A}\,\varphi & [\textsc{All Futures}] & & & \\
\mid & \mathtt{E}\,\varphi & [\textsc{Some Future}] & & &
\end{array}$$

Fig. 13: STLCS syntax

Besides classical Boolean connectives, and the operators of SLCS we have introduced in Sect. 4, STLCS 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 a path-specific operator, namely $\mathcal{X}$ ("next"), `F` ("eventually"), `G` ("globally"), $\mathcal{U}$ ("until").

The definition of a model $\mathcal{M}$ of STLCS is based on the notion of Kripke frame (see Definition 16) and is given below.

Definition 23. *A model is a structure $\mathcal{M} = ((X, \mathcal{C}), (S, \mathcal{R}), \mathcal{V}_{s\in S})$ where $(X, \mathcal{C})$ is a quasi-discrete closure space, $(S, \mathcal{R})$ is a Kripke frame, and $\mathcal{V}$ is a family of* valuations*, indexed by S; for each $s \in S$, we have $\mathcal{V}_s : AP \to \wp(X)$.*

The truth value of a formula is defined at a point in space x at state s. Valuations of atomic propositions depend both on states and points of the space. Intuitively, there is a set of possible worlds, i.e. the states in S, and a spatial structure represented by a closure space. In each possible world there is a different valuation of atomic propositions, inducing a different "snapshot" of the spatial situation which "evolves" over time (non-deterministically); see Figure 14 (left) where space is a two-dimensional structure, and valuations at each state are depicted by different colours. In this tutorial we assume that the spatial *structure* $(X, \mathcal{C})$ does not change over time. However, other options are possible. For instance, when space depends on S, one may consider an S-indexed family $(X_s, C_s)_{s \in S}$ of closure spaces.

A *path* in the model is a sequence of *spatial models* indexed by instants of time; see Figure 14 (right). Given that we assume that the closure space $(X, \mathcal{C})$ does not depend on S, in the sequel we will use a simplified notion of path, defined on Kripke frames, instead of on full models. In other words, we consider paths of indexes (cf. Definition 23).

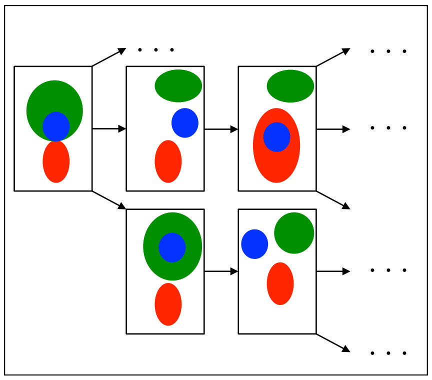
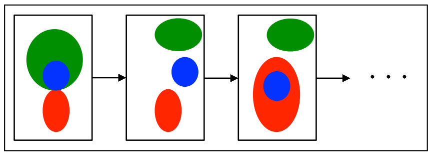



Fig. 14: In spatio-temporal logics, a temporal structure represents a computation tree of *snapshots* induced by the time-dependent valuations of the atomic propositions (left). A path in the model (right).

Definition 24. *Given Kripke frame $\mathcal{K} = (S, \mathcal{R})$, a* path *$\sigma$ is a function from $\mathbb{N}$ to S such that for all $n \in \mathbb{N}$ we have $(\sigma(i), \sigma(i+1)) \in \mathcal{R}$. Call $\mathcal{P}_s$ the set of infinite paths in $\mathcal{K}$* rooted *at s, that is, the set of paths σ with $\sigma(0) = s$, where, whenever for some i there is no $s' \in S$ s.t. $(\sigma(i), s') \in \mathcal{R}$, we let $\sigma(j) = \sigma(i)$ for all $j > i$ (self-loop completion).*

The evaluation contexts are of the form $\mathcal{M}, x, s \models \Phi$, where Φ is a STLCS formula, s is a state of a Kripke frame, and x is a point in space X. The formal semantics of the logic is given in Definition 25 and can also be found in [16].

Definition 25. *Satisfaction is defined in a model* $\mathcal{M} = ((X, \mathcal{C}), (S, \mathcal{R}), \mathcal{V}_{s \in S})$ *at point* $x \in X$ *and state* $s \in S$ *as follows:*

$$
\begin{array}{lcl}
\mathcal{M}, x, s \models \top & & \\
\mathcal{M}, x, s \models p & \iff & x \in \mathcal{V}_s(p) \\
\mathcal{M}, x, s \models \neg\Phi & \iff & \texttt{not}\, \mathcal{M}, x, s \models \Phi \\
\mathcal{M}, x, s \models \Phi \wedge \Psi & \iff & \mathcal{M}, x, s \models \Phi \texttt{ and } \mathcal{M}, x, s \models \Psi \\
\mathcal{M}, x, s \models \mathcal{N}\Phi & \iff & x \in \mathcal{C}(\{y \in X | \mathcal{M}, y, s \models \Phi\}) \\
\mathcal{M}, x, s \models \Phi \mathrel{\mathcal{S}} \Psi & \iff & \exists A \subseteq X . x \in A \texttt{ and } \forall y \in A . \mathcal{M}, y, s \models \Phi \wedge \\
 & & \qquad \texttt{and } \forall z \in \mathcal{B}^{+}(A) . \mathcal{M}, z, s \models \Psi \\
\mathcal{M}, x, s \models \texttt{A}\, \varphi & \iff & \forall \sigma \in \mathcal{P}_s .\, \mathcal{M}, x, \sigma \models \varphi \\
\mathcal{M}, x, s \models \texttt{E}\, \varphi & \iff & \exists \sigma \in \mathcal{P}_s .\, \mathcal{M}, x, \sigma \models \varphi \\
 & & \\
\mathcal{M}, x, \sigma \models \mathcal{X}\Phi & \iff & \mathcal{M}, x, \sigma(1) \models \Phi \\
\mathcal{M}, x, \sigma \models \Phi \mathcal{U} \Psi & \iff & \exists n .\, \mathcal{M}, x, \sigma(n) \models \Psi \texttt{ and } \forall n' \in [0, n) . \mathcal{M}, x, \sigma(n') \models \Phi
\end{array}
$$

where $[n, n) = \emptyset$ *for all* $n \in \mathbb{N}$*.*

Let us proceed with a few examples. Consider the STLCS formula

$$\texttt{E}\,\texttt{G}\,(\mathit{green}\,\mathcal{S}\,\mathit{blue})$$

Point x satisfies such formula in state s if there exists ($\texttt{E}$) a temporal path rooted at s, such that in all states ($\texttt{G}$), i.e. at any point in time, x satisfies atomic property *green*, and it is not possible to start from x, following edges of the spatial graph, and leave the region of points satisfying *green* in which x is located, unless passing by a point satisfying *blue*.

The mutual nesting of spatial and temporal operators permits one to express rather complex spatio-temporal properties. An example exhibiting nesting of spatio-temporal operators is the STLCS formula

$$\texttt{E}\,\texttt{F}\,(\mathit{green}\,\mathcal{S}\,(\texttt{A}\mathcal{X}\,\mathit{blue}))$$

This formula is satisfied by a point x in state s if point x possibly ($\texttt{E}$) satisfies *green* in some future ($\texttt{F}$) state s', and in that state, it is not possible to leave the area of points satisfying *green* unless passing by a point that will necessarily ($\texttt{A}$) satisfy *blue* in the next ($\texttt{X}$) time step.

7 Spatio-temporal Model Checking

Based on the formal semantics of the spatio-temporal logic STLCS, presented in the previous section, `topochecker` has been extended to deal with spatio-temporal properties. In the spatio-temporal setting, models are composed of a temporal part, which is a Kripke frame, and a spatial part, which is a finite, quasi-discrete closure space.

The model checker enriches basic STLCS by allowing users to use floating-point variables and define some atomic propositions as simple assertions on the value of such variables, e.g. comparison of such values with (floating-point) constants. It finally permits parametric macro abbreviations, that we use in the examples in the next section.

The temporal part of the model checking algorithm is a variant of the well-known Computation Tree Logic (CTL) labelling algorithm, whereas the spatial part is based on the algorithms described in Sect. 5. For more information on CTL and its model checking techniques, see e.g., [4] or [21]. Given a formula Φ and a model $\mathcal{M}$, the algorithm proceeds by induction on the structure of Φ; the output of the algorithm is the set of pairs (x, s) such that $\mathcal{M}, x, s \models \Phi$. For further details of the algorithm we refer to [16].

The complexity of the 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 checking algorithms when the spatial part of the model is relatively small, as shown in the examples in the next section. Further improvements of the efficiency is part of future work.

To illustrate spatio-temporal model checking, let us continue the example of the wave pattern formation introduced in Section 4. We can use spatio-temporal formulas to study the evolution of the wave pattern over time. As a model we use a sequence of snapshots of the first 100 time steps of the solution of the equations (8) of Section 4. Such a sequence is a simple Kripke frame without any non-determinism. The last snapshot is repeated in an artificial way to obtain an infinite path. We can use this structure, for example, to identify which points are part of a pattern for a number of consecutive steps in the evolution. We start from the spatial property 'pattern' as in Section 4:

$$\texttt{pattern} = [\texttt{a<0}]\texttt{S}[\texttt{a>=0}]$$

Then we define the various periods, ranging from 3 to 10 time steps, during which a point remains part of the pattern as follows (using the frontend notation of `topochecker` also for STLCS formulas):

$$\begin{aligned}
&\texttt{pattern2steps} &&= \texttt{pattern} \,\&\, \texttt{A X} (\texttt{A X} \, \texttt{pattern})\\
&\texttt{pattern3steps} &&= \texttt{pattern2steps} \,\&\, \texttt{A X} (\texttt{A X} (\texttt{A X} \, \texttt{pattern}))\\
&\cdots\\
&\texttt{pattern10steps} &&= \texttt{pattern9steps} \,\&\, \texttt{A X} (\texttt{A X} (\texttt{A X} (\cdots (\texttt{A X} \, \texttt{pattern})) \cdots)
\end{aligned}$$

The spatio-temporal model checker can verify multiple properties simultaneously and show their results in different selected colours[12]. Note that the properties require a point to satisfy a particular spatial property to hold in the current snapshot and in several subsequent snapshots. Moreover, we can verify

[12] Note that the results may involve the same points, in which case the later result overwrites the previous result.

such properties starting from the initial snapshot, but also starting from any other chosen snapshot.

Figure 15 shows the evolution when the formulas are evaluated taking as initial snapshot the one at time 10, 20, 30 and 40, respectively. The results show that the pattern seems to stabilise starting from the north-western corner of the figure after which the points towards the south-eastern corner become increasingly stable, at least for 10 subsequent steps in time.

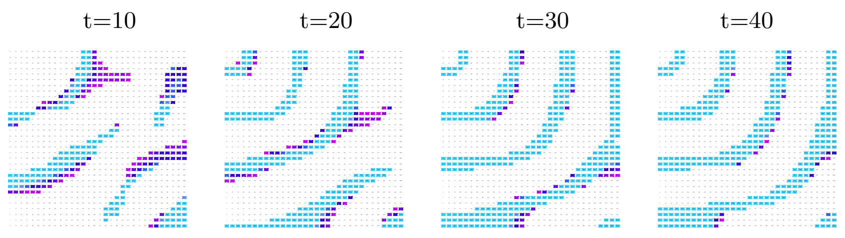


Fig. 15: Formation of a wave-like pattern; evolution in steps of 10; pink points are part of a wave for 2 subsequent steps; cyan points for 10 subsequent steps. Other colours represent the intermediate number of steps.

We can also check the emergence of truly stable points, i.e. points that, once they satisfy property `pattern`, they continue to do so until the end of the simulation trace. This property can be formalised as:

$$\texttt{pattern\_permanent} = \texttt{pattern} \,\&\, \mathtt{A}\,\mathtt{G}\,\texttt{pattern}$$

The model checking results for this formula are shown in Figure 16. Points that satisfy property `pattern_permanent` are orange.

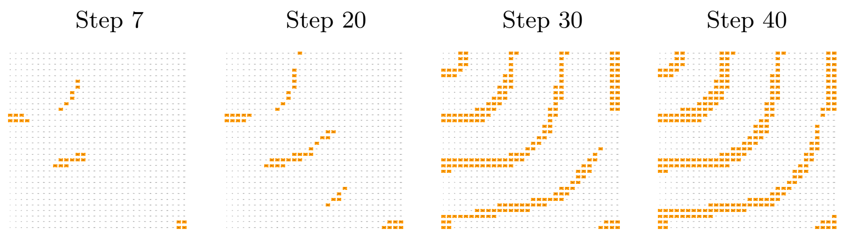


Fig. 16: Formation of a wave-like pattern; formula `pattern_permanent` evaluated starting from snapshots at times 7, 20, 30 and 40 of the simulation trace, respectively.

8 Case Studies on Collective Adaptive Systems

We present several case studies to illustrate the use of spatio-temporal model checking based on closure spaces for Collective Adaptive Systems. The first case study concerns emergent behaviour in bike sharing systems. The case study builds on earlier joint work [35,20] involving some of the co-authors. The system model, based on Markov Renewal Processes, explicitly takes a part of user behaviour into account that is usually not visible in the data collected by operators of bike sharing systems, such as failure to obtain a bike when needed or difficulties to park a bike close to the desired destination. We illustrate how spatio-temporal model checking can be used to obtain a deeper insight into these problems, using a model-based approach, that complements other analysis methods.

In this first example the structures used for model checking are linear in nature. This is due to the fact that they are produced as single simulation traces of a complex behavioural model. In the second case study a richer structure is addressed in which branching time behaviour is combined with spatial aspects. Such more complex structures reflect non-deterministic choice between possible ways the behaviour may evolve in time and space and in this case this is exploited to analyse the effect of adaptation strategies in the context of public bus transportation systems. The example extends the work on spatial model checking applied to public bus transportation systems presented in [15].

Both examples use `topochecker` as supporting tool. Consequently, also in this section we use the tool's frontend syntax for the logic formulas. The correspondence to the languages defined in Section 4 and Section 6 is straightforward.

8.1 Bike Sharing

Smart bike sharing systems (BSS) have recently become a popular public transport mode in hundreds of modern cities [22,36] operating from a few (e.g. Pisa) up to hundreds or thousands of docking stations (e.g. Hangzhou, Paris, or London[13]). The principle of bike sharing is quite simple. A number of stations with docks partially filled with bicycles are placed throughout a city. Users of the service may hire any bicycle at any station at any time for private use, and must return it at some station of their choice. The initial period of, typically, thirty minutes is free of charge, after which an hourly fee is charged. The operator assumes the responsibility to maintain a high level of usage of the system.

Operating a BSS raises multiple issues such as efficiency of fuel-consuming repositioning services [24], or integration with other public transport modes. Not the least, it is important to make the service attractive to its users. An indication that user satisfaction should be addressed seriously can be found in a survey of user experience with the *Bicing* BSS in Barcelona, conducted by Froelich et

[13] Pisa: `http://www.pisamo.it`, Hangzhou: `http://www.publicbike.net`; Paris: `http://www.velib.paris.fr`, London: `https://tfl.gov.uk/modes/cycling/santander-cycles`

al. [25]. It reports that 75% of the users who used it for commuting between their home and study or work, stated that 'finding an available bike and parking slot' were the two most important problems, encountered in 76% and 66% cases of 212 respondents, respectively. Since a bicycle must be returned to one of the designated stations, otherwise a high fine needs to be paid, users must find a drop off station that is not full. The additional searching time, and the associated risk of undesired and unpredictable delays, and fees, are likely to affect the user's satisfaction with the system. However, user satisfaction due to such temporarily full stations is difficult to evaluate quantitatively when only data obtained from real systems is available. This is so because the cycling data alone is not sufficient to understand the *intentions* of its users, nor the predictability of the service. To investigate this issue from a different angle, a model-based approach was presented in [35], in which the point of view of 'rational agents' who participate in bike-sharing, is assumed. The results of the study reveals that cycling times in different cities, and among pairs of stations within cities, are similarly distributed[14], suggesting a possibility of a generic interpretation. The rational agent model, based on minimal assumptions about travelling and decision making, reproduces rather well the cycling time distributions in London and Pisa [35]. The analysis suggests that some features of the cycling time distribution may be related to the (un)predictability of a travel process which, as just discussed, may influence the users' satisfaction with a system. In particular, the results suggest a further analysis extending the notion of a problematic full station to the notion of a problematic *area* in which all stations are full: a full-station *cluster*. Formation of clusters and their evolution is also evident from the existing bike-sharing visualisations[15].

The main advantage of a modelling approach over data analysis is a possibility to study hypothetical cases where station configurations, traffic flows, or incentives are altered to explore the efficiency of proposed solutions to the aforementioned issues. In this example we will analyse some traces generated by a simulation model developed in [35] using `topochecker`.

The examples presented here and some further examples can be found in [20]. We are concerned with the full-station clusters as more salient to the discussion, although the extension of the same kind of formulas to clusters of empty stations is straightforward.

The bike-sharing model presented in [35] describes the dynamics of a population of rational agents, coupled to the dynamics of bicycle stations in a two-dimensional rectangle representing a city. The rational agent model is based on a Markov Renewal Process (MRP [35]). The model parameters, such as the number of stations, cycling pace and request rate, are calibrated so as to reproduce cycling times of a particular real BSS, in this particular case that of London. The result is a $7{\times}13$ km^2 area with a $19{\times}38$ array of stations with randomly perturbed locations, random capacities between 15 and 40 docks, and 500 agents that make, on average, 900 trips per hour. The agent behaviour is sampled ran-

[14] See also [10]

[15] See, e.g. `http://bikes.oobrien.com/london`

domly. However, to introduce flows, a superposition of Gaussian distributions for the origin and destination locations is used and some counter-current flows are added to improve the balance of the flow. Numerical simulation of this model generates traces, each trace consisting of snapshots, each snapshot representing a system's state at a particular instance of time.

Agents risk, of course, that a suitable station within the area is not found. These 'bad' events affect only a small fraction of all trips if the distribution of agents' origins and destinations is spatially homogeneous, but become more relevant if some destinations are more popular than others. Presence of areas that attract more users than other areas is a reasonable assumption about real cities. An obvious consequence is that also the areas of full stations will be, as a rule, larger. However, identification and analysis of problematic areas with traditional means is not so straightforward as they dynamically evolve over time.

Spatial structure is added to the simulation model in the form of an undirected graph, whose vertices represent stations, and edges represent the nearest station connections. The graph represents 722 stations, arranged in a grid layout of 19×38 nodes as explained above. A single trace of the simulation model is used as input to the spatio-temporal model-checker. It represents the evolution of the model at specific time intervals, for a given number of steps. It provides for each station, at each step, the number of bikes parked in it, the number of free parking places and its capacity (among other information). For all the experiments described below, except the last one (related to user satisfaction), the duration of an interval is 10 minutes, and the number of time steps is 101. In the last experiment, we considered a time interval of 1 minute and 301 time steps[16]. Starting with simple expressions of a system's state, we proceed to develop more complex formulas that nest spatial and temporal operators.

Full stations and clusters. First, we characterise stations that are *full*, that is, with no vacant places, and *clusters* of full stations, that is, stations that are full, and are connected only to other stations that are full in turn. These two (purely spatial) properties are formalised below.

$$\begin{aligned} \texttt{full} &= [\texttt{vacant}{==}0] \\ \texttt{cluster} &= \texttt{I}\,(\texttt{full}) \end{aligned}$$

The macro abbreviation `full` uses a boolean predicate (equality), applied to the quantitative value of the atomic property [`vacant`]. Connectivity is expressed by the derived *interior* operator $\texttt{I}\,\Phi = \texttt{!}\,(\texttt{N}\,(\texttt{!}\,\Phi))$. Informally speaking, in an undirected graph, points satisfying $\texttt{I}\,\Phi$ are only connected to points satisfying Φ. The smallest possible cluster is therefore composed of a full station such that its direct neighbours in the north, south, east and west directions are also full. Note that the definition of `cluster` only identifies (on purpose) these "inner" full stations and not their direct full neighbours.

[16] The results can be reproduced using the data and scripts, provided with the source code of the tool.

Formation of clusters. A point evolves into a cluster when it becomes full, and stays full until it becomes part of a cluster. This may be detected by the following formulas:

$$\begin{aligned}\texttt{implies}(\texttt{f},\texttt{g}) &= (!\,\texttt{f}) \mid \texttt{g};\\ \texttt{nextCluster} &= (\texttt{E}\,\texttt{F}\ \texttt{full})\ \& \\ &\qquad (\texttt{A}\,\texttt{G}\ \texttt{implies}(\texttt{full}, \\ &\qquad\quad \texttt{A}\ \texttt{full}\ \texttt{U}\ \texttt{cluster}))\end{aligned}$$

Here, `implies` is standard logical implication. The definition of `nextCluster` characterises points that will eventually become full and, for every future state, whenever full, they will remain full until becoming part of a cluster. This is a very strong property, that few points possess. Such points are central in cluster formation, as they represent stations that always form a cluster when they become full. In Figure 17, these points are shown in red, in a state[17] of the simulation where there are many of them. For comparison, the boundary of the points that will become a cluster are shown in green, that is, those points satisfying (`N EF cluster`) & (`! EF cluster`).

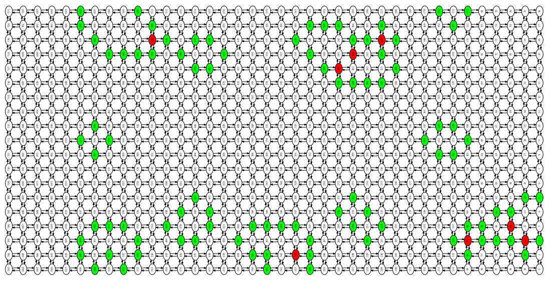


Fig. 17: Stations of time step 80 of the simulation that are on the boundary of the region of points that will eventually become a cluster (green), and stations that, whenever they are full, stay full and become part of a cluster (red).

Persistence of clusters. We can identify stations belonging to clusters that *persist* for some amount of time, that is, they last for a specific number of time steps. This situation is described for two and three time steps by the formulas:

$$\begin{aligned}\texttt{cluster2steps} &= \texttt{cluster}\ \&\ (\texttt{A}\,\texttt{X}\ \texttt{cluster})\\ \texttt{cluster3steps} &= \texttt{cluster}\ \&\ (\texttt{A}\,\texttt{X}\ \texttt{cluster2steps})\end{aligned}$$

By combining the formulas described above with the *eventually* operator, the tool is able to detect the stations that, in any state, will eventually be

[17] The tool is a *global* model checker, therefore it is able to produce a graph for each state of the model, related to the truth value of formulas in that particular state, even if we only show results related to one specific state.

part of a cluster, with specified persistence. Let us look at Figure 18, where we show the output of a model checking session. The tool colours in red nodes that satisfy the formula `EF cluster3steps` (and thus also formulas describing shorter persistence times), in blue those that satisfy `EF cluster2steps`, and in green those where formula `EF cluster` is true.

Propagation of clusters. Another phenomenon that can be investigated using `topochecker` is the spatial propagation of clusters. Among many possible related STLCS formulas, we show how to detect points that satisfy at least one of the following: 1) they are not full, but are close to a cluster, and will necessarily become part of a cluster in the near future (`growingCluster`); or 2) they are part of a cluster, but will necessarily become not full in a short amount of time, even if still being physically close to a cluster (`shrinkingCluster`). This is achieved by the following definitions, where the macro `bdry` implements the derived 'boundary' operator δ (see Fig 6).

```
bdry(f)          = (N f) & (!f)
growingCluster   = (!cluster) &
                    (N (bdry(cluster)) & (A X full)
shrinkingCluster = cluster & (A X (!full))
```

For instance, in time step 77 of our simulation, there are both stations that will join a cluster and stations that will leave a cluster in the next time step. We show the result in Figure 19 using different colours for facilitating comparison of results. The stations that satisfy `cluster` are green, the stations satisfying `growingCluster` are red and stations satisfying `shrinkingCluster` are blue. The results of these formulas provide insight in the dynamics of the clusters at particular time steps, in particular the directions in which the clusters are evolving. This may be important information for the development of repositioning strategies in particular when such dynamics are repeated over time in the same areas.

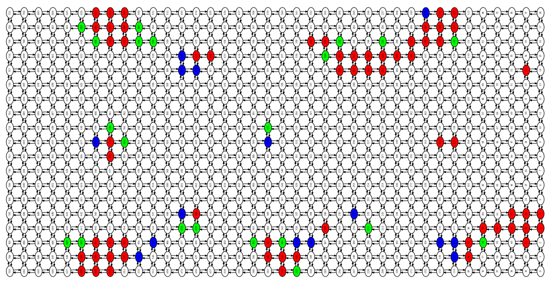


Fig. 18: Points of the initial state of the simulation that will eventually become part of a cluster (green), or of a cluster that persists for two (resp. three) time steps, coloured in blue (resp. red).

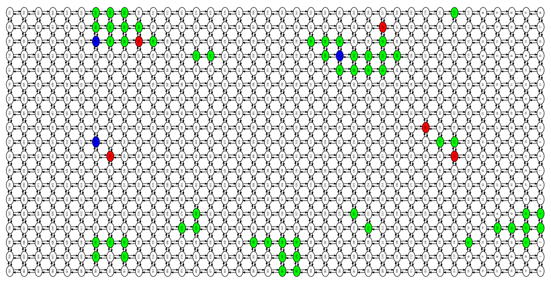


Fig. 19: Points of time step 77 of the simulation that are part of a cluster (in green), or are not full, but will become part of a cluster in one step (in red), or that are part of a cluster but will become not full in one step (in blue).

User experience. STLCS can also be used to identify specific problems related to user experience in BSSs. For example, when a user wants to leave a bike at a specific station, and that station is full, she may try to find a nearby station with available parking slots, or she may wait for some time in the same station. This behaviour may be typically sufficient to solve the problem, at the expense of a longer trip duration. One may want to check whether this procedure is effective in a few time steps. In the following formula, we check whether it is possible that, in three time steps, the user still was unable to leave the bike in the same or a nearby station, when the preferred station is full in the current state.

$$\begin{aligned}\texttt{tripEnd} = \texttt{full} \,\&\, (\texttt{N}\,(\texttt{A}\,\texttt{X}\,(\texttt{full} \,\&\, \\ (\texttt{N}\,(\texttt{A}\,\texttt{X}\,(\texttt{full} \,\&\, \texttt{N}\,(\texttt{A}\,\texttt{X}\ \texttt{full})))))))\end{aligned}$$

Figure 20 shows the output from the model checker, where the red points indicate the stations where the formula is true at time step 0. The formula can be checked for various numbers of consecutive time steps, which provides an indication of how severe the problem is at a particular time of interest and a particular area. Ideally, the formula should not hold anywhere, or at most be true only in a few points and for a small number of steps.

8.2 Bus Clumping in Frequent Bus Services

The next example concerns the analysis of some specific problems in modern public transport systems such as the so-called "frequent" bus services operating in many densely populated cities. Frequent bus services are public bus services without a published timetable but with regular and frequent buses operating along pre-established routes. In such services a particular phenomenon may occur that is commonly known as *bus bunching* or *bus clumping*. Bus 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

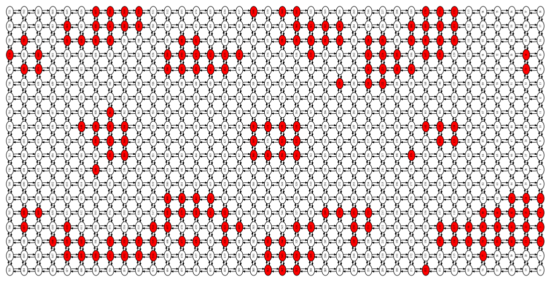


Fig. 20: Stations that are full in time step 0, where it is possible that an user applying the obvious strategy of waiting for some time, and then moving nearby, might still not find a free parking slot.

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. The problem is illustrated in Figure 21 showing three successive "satellite view" images of the position of three different buses on the same route projected on a portion of a map of a city. The buses are shown as three small red dots on the main road. The distance between the buses is getting smaller in each successive image, resulting in the three buses being lined up in the final image as indicated by the red arrows (Figure 21 left-bottom).

In modern public transport systems the operation depends on accurate fleet management which is supported by an automatic vehicle location (AVL) system. Since these data are used in real-time in other systems such as bus arrival prediction systems, it is very important that these data are checked for correctness. In [15] it was shown how spatial model checking can be used to check for a number of typical error conditions occurring occasionally in the data. In this approach, the data is mapped onto a digital map of a city, much like the one shown in Figure 21, after which spatial model checking is applied on the augmented map to identify the error conditions in an automatic way. Among the typical error conditions are bus positions that are off the road, for example in a field or in water, that indicate possible errors in the transmission of the GPS data, or bus positions that are away from the planned route, which may indicate problems with the road network.

In joint follow-up work with other authors [14] we focussed on the analysis of adaptive correction strategies to mitigate the occurrence of clumping and providing better service to the public. One such strategy is based on sending a bus the instruction to wait for a short time when getting too close to the bus ahead. Such wait-requests should be only suggested by the service operators and not imposed, since there may be several circumstances in which a bus is not in the position to be able to wait, for example due to specific traffic circumstances. Also the number of wait-requests should be carefully considered. For example, if

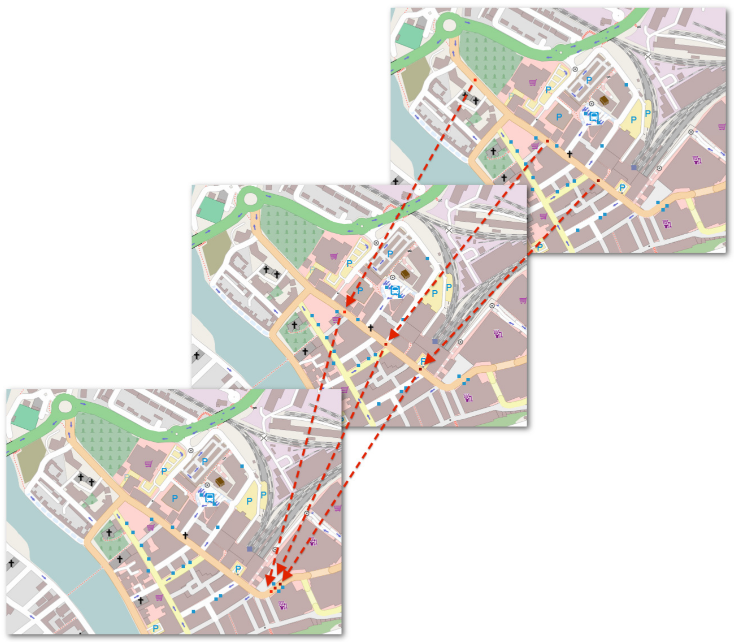


Fig. 21: Because of delays caused by boarding passengers the headway between buses is successively eroded over time until the buses are essentially ‘clumped’ together.

all buses on a route were sent a wait-request, and all accepted this request, then this would not solve the problem. Similarly, sending multiple wait-requests to the same bus may create unacceptable delays. Apart from these general heuristics, there are still many options for possible strategies. In the following we illustrate how spatio-temporal model checking, based on a branching time temporal logic, can be of use to analyse the effect of such correction strategies.

First we show how spatio-temporal logic can be used to detect clumping of buses in a system trace. Such a trace consists of GPS data of a number of buses operating on the same route projected onto a digital city map. After that we show how this characterisation of clumping can be used in combination with a constructed *branching* model that captures the various ways in which wait-requests can be issued and their effect on bus positions.

Spatio-temporal characterisation of clumping Consider a single bus route, served by k buses. At each instant of time, the state of the system is completely described by a tuple of k GPS positions; therefore, a system trace is a finite sequence of such tuples. We can distinguish two different variants of clumping that differ in a subtle way. One in which two consecutive buses serving the same route are *spatially* close to each other, and one in which they pass by the same bus stop within a too short amount of time. Here we formalise the latter variant considering three buses on the same route. The input code of the model checking session is shown in Figure 22. Formulas `bus1`, `bus2`, `bus3`, and `busStop`, are defined as colour ranges, that serve the purpose of identifying bus positions on a digital map of the city. In this example, colours are used to distinguish the different buses serving the same route, so that each bus has a specific colour. Similarly, formula `busStop` identifies the position of a bus stop. The formula `timeConglomerate`, that we explain below, is true at points of a *bus stop* whenever clumping is happening (formation of a conglomerate of buses) at that particular stop.

```
bus1 = <RED [155,155]> & <GREEN [0,0]> & <BLUE [0,0]>;
bus2 = <RED [188,188]> & <GREEN [0,0]> & <BLUE [0,0]>;
bus3 = <RED [221,221]> & <GREEN [0,0]> & <BLUE [0,0]>;
bus = bus1 | bus2 | bus3;
busStop = <RED [55,55]> & <GREEN [55,55]> & <BLUE [255,255]>;

close(x) = N^7 x;

busAtStop(x) = busStop & close(x);

busAfterBus1 = busAtStop(bus1) &
     EX busAtStop(bus2 | bus3);
busAfterBus2 = busAtStop(bus2) &
     EX busAtStop(bus1 | bus3);
busAfterBus3 = busAtStop(bus3) &
     EX busAtStop(bus1 | bus2);

timeConglomerate = (busAfterBus1 | busAfterBus2 | busAfterBus3);
```

Fig. 22: Spatio-temporal formulas for time conglomerates

A *spatio-temporal* conglomerate happens when two buses serving the same route pass by the same stop within a short amount of time. This case is subtler than the spatial one, as it does not necessary imply that the headway between two buses becomes too small. This event is described by the formula `timeConglomerate`, which features a combination of spatial operators (used to detect that a bus is close to a stop) and temporal operators (used to identify the spatio-temporal conglomerate). For instance, consider the formula `busAfterBus1`. This formula is true on points that are: i) part of a bus stop, and close to `bus1`, because `busAtStop` must be true for `bus1`; ii) such that, in the next snapshot[18], these will be part of a bus stop, and close to either `bus2` or `bus3`. Note that the

[18] More than one time step can be required. This can be achieved by repeated nesting of the `EX` operator. We did not do so for the sake of clarity in Figure 23.

use of spatial and temporal connectives in the same formula permits one to refer to the colour of points at a specific time, and at subsequent time instants.

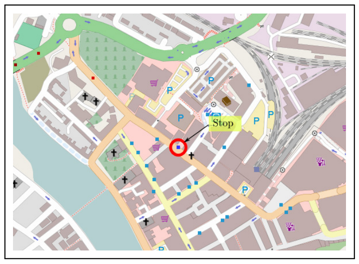
Figure 23 is obtained from the spatio-temporal model checker, starting from the positions of three buses serving the same route. Figures 23a-23e are obtained by mapping bus coordinates over a base map. Buses are represented by small squares of different shades of red on the roads. To make them more visible they are also highlighted by the red circles in Fig. 23b. The small dark blue square is a bus stop (see Fig. 23c). Figure 23f shows the output of the model checker when checking the formula `EF timeConglomerate` in the initial state shown in Fig. 23a. Indeed, Figure 23f is the same as Figure 23a, except for the colour of the bus stop, whose points are now turned green by the model checker, indicating that clumping happens at that stop, at some point in the future.

The effect of correction strategies Let us now address the effect of correction strategies to mitigate clumping. In particular, we use existing data (e.g. system logs) in estimating the impact of introducing new policies in a system. The spatio-temporal model checker for STLCS can be used both to detect clumping in a single system trace, as we have seen, and to analyse a *branching* model, that is, a system where at each state, non-deterministically, there may be several possible steps to different future states. Such non-deterministic models represent in a concise way a great number of possible system behaviours, depending on the choices that may be made at each execution step. We use this fact in conjunction with the idea to send buses wait-requests in order to reduce clumping. The possibility of issuing wait-requests to specific buses, or not doing so, introduces non-deterministic choice points, where some buses go ahead and others are kept on hold for a short period of time.

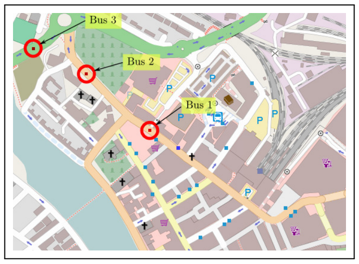
In more detail, consider a system trace of AVL data (e.g., provided by the bus company[19]). Let us assume that this trace reflects a typical period of the day in which clumping of buses often occurs and that the trace contains the position of the relevant buses at fixed small intervals of time of say 30 seconds. The trace is a sequence of tuples of k elements, where k is the number of buses serving the same route. The length of the sequence is the number of samples. Element at position i in each tuple is the position of bus i. At each step, besides the already existing transition to the next step, more transitions are added, to new states, where one or more buses wait, (therefore, their position does not change), and the other ones move as they actually did in the original system trace that was taken as starting point.

In order to illustrate the approach, a transformation algorithm was implemented. The implementation is parametric with respect to the maximum number of buses that are allowed to wait simultaneously, and the maximum number of wait instructions issued to the same bus. The input of the algorithm consists of the system trace as described above, and of a map. The state space of the branching model generated from the system presented in Figure 23 is shown in

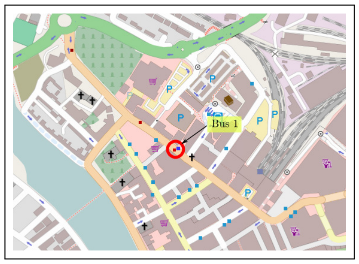
[19] We use artificial data for the sake of simplicity, but usage of the approach does not differ on real data.



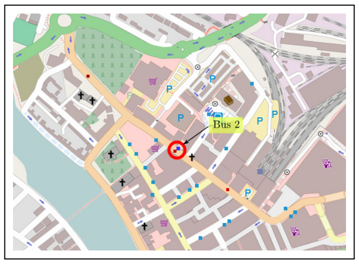
(a) Initial state



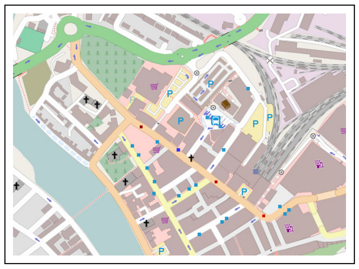
(b) Second state



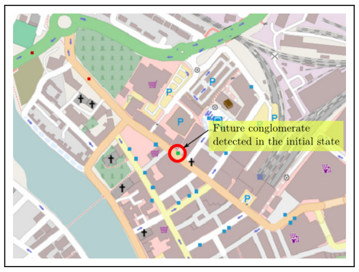
(c) Bus 1 passes by the stop



(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

Fig. 23: Spatio-temporal conglomerate.

Figure 24; as typical in CTL model-checking, self-loops have been added to terminal states, since CTL-path formulas express properties of infinite paths, i.e. infinite sequences of states. To verify that there exist traces in which no conglomerate occurs, `topochecker` is applied on the formula `AF timeConglomerate` and the branching model. The formula is true only if a conglomerate is present on *all* (infinite) system paths. In this case no point is coloured, meaning that there exist "good" paths for each point in the model, and thus waiting-strategies that could avoid clumping in the situation represented by this trace. If the trace is indeed representative of a daily recurring situation, the conjecture is that the waiting strategies have a good chance to mitigate the clumping problem also on other days during similar periods of time.

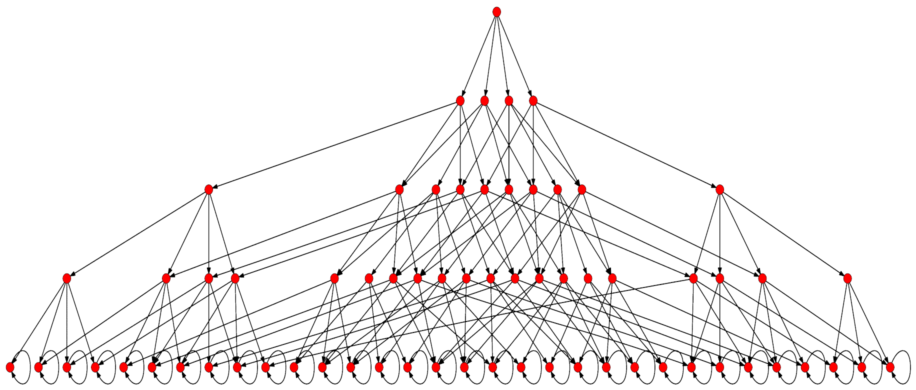


Fig. 24: Branching model obtained by augmenting a linear trace.

It is not difficult to add a facility to the model checker to generate counter examples. This can be used to obtain in an automatic way the traces that represent a waiting-strategy that avoids the emergence of clumping for the considered situation. In a second phase one could compare the quality of the different strategies and, for example, select those with the minimal number of waiting-requests.

The simple construction to consider all possible waiting strategies could be made more sophisticated by not considering a choice in *every* state of the trace, but considering choices only when a situation is detected that is known to lead to clumping in the near future with a high probability. This could be obtained, for example, using machine learning techniques. This would reduce considerably the number of states to be considered.

9 Outlook and Future Work

In this tutorial we have provided an introduction to spatial and spatio-temporal logics and model checking for closure spaces. Formal verification for continuous and discrete spatial models is still a relatively unexplored area of research. Let us conclude by briefly reviewing recent and ongoing work and open issues for the particular approach that we have addressed in this tutorial.

Further Logical Operators Besides the basic operators and the many useful derived operators that have been presented in this tutorial, the list of operators could be further extended. Of course, this should be done in a careful way keeping in mind that the basic operators are really independent, i.e. they cannot be derived from existing operators, but also such that the properties involving these operators can be efficiently verified. Ideally, the operators have a similar semantics when interpreted on both continuous and discrete spatial models.

Along these lines, in forthcoming work [19] we investigate several new operators. The first operator $\mathcal{P}$ is capturing the notion of *propagation* and is inspired by the duality in the direction in which spatial operators can be applied. Informally, a point x satisfies $\varphi_1 \, \mathcal{P} \, \varphi_2$ if it satisfies φ_2 and it is *reachable from* a point satisfying φ_1 via a path such that all of its points, except possibly the starting point, satisfy φ_2. For instance, if we consider again the model of Figure 2, *blue*, *green* and *white* nodes satisfy $\mathit{green} \, \mathcal{P} \, \neg \mathit{red}$ while the same formula is not satisfied by *yellow* nodes.

Furthermore, it is useful to introduce spatial operators that predicate on one or more *sets of points* rather than individual points. For example, we might want to know whether points with a particular property are actually part of a larger set of points satisfying another property, but being all connected to one another. Thus, for instance, this way one could verify whether three buses are actually on the same street segment, or whether some agents are in the same region of a maze from which they can reach the same exit.

Further extensions, involving (maybe) two sets of points, could lead to an alternative formulation of the region calculus in the setting of closure spaces.

Extensions of Spatio-temporal model checking There are various ways in which spatio-temporal model checking can be extended. First of all, further optimisations of the model checking algorithms should be investigated. In particular, in many models the spatial evolution over time could be gradual and not involve all points of the space, or the spatial evolution could be 'predictable' to some extent in specific classes of behaviours, which may allow for further optimisations.

In recent work the spatial surround operator has also been added to a different temporal logic, namely the signal temporal logic [37,38], which in turn is an extension of the Metric Interval Temporal Logic [34]. The Signal Temporal Logic is used to reason about continuous signals. Its extension with spatial modalities provides a linear time spatio-temporal logic. The two modalities are the *bounded somewhere* $\diamondsuit_{[w_1,w_2]}$ and the *bounded surround* operator $\mathcal{S}_{[w_1,w_2]}$, an extended adaptation of the spatial until operator of [18] to the signal models framework. 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 (or distance) greater than or equal to w_1 and less than or equal to w_2. The surround formula $\varphi_1 \; \mathcal{S}_{[w_1,w_2]} \, \varphi_2$ 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.

The *everywhere* operator can be derived as the dual of the somewhere 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 (or distance) between w_1 and w_2.

The logic has both a boolean semantics and a quantitative semantics. The former defines when a formula is satisfied, the latter provides an indication of the robustness with which a formula is satisfied [6,7,5], i.e. how susceptible it is to changing its truth value, for example, as a result of a perturbation in the signals. SSTL is interpreted on spatio-temporal, real-valued signals. Such 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 [37] 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. Efficient monitoring algorithms have been developed for SSTL and implemented in a prototype spatio-temporal model checker.

Spatio-temporal model checking can also be conceived for models with stochastic time. For example, in the case study of the London bike sharing system we could be interested to know how likely it is that a station becomes part of a cluster of full stations. We are currently developing a statistical model checking approach for such probabilistic spatio-temporal properties. Another approach could be to extend the spatio-temporal logic itself with forms of stochasticity. There are different options. One is to add stochasticity to time, another to add it to space, a third option would be to add stochasticity to both. Furthermore, as we have seen in the Turing pattern examples, partial differential equations can be used to define large spatial collective systems featuring interaction and mobility. In some CAS it may be of interest to study the behaviour of a single individual in the context of a large spatial collective system. This can be done using for example fluid and mean field model checking [12,11,33]. Combining spatial model checking with such mean field model checking approaches would be very interesting but represents also a great challenge because of issues of scalability of the approach, in particular for what concerns the spatial aspects.

Path-based definition of SLCS logics As we have seen in Section 4, Proposition 5 allows us to reformulate the definition of the $\mathcal{S}$ using the notion of path. In other words, one could take the characterization of $\mathcal{S}$ as in Proposition 5 and use it as a definition; in the context of *quasi-discrete* closure spaces, the latter would be equivalent to Definition 22.

Interestingly, the very same, path-based, definition provides more intuitive results when interpreted on topological spaces, which instead is not the case when using Definition 22 for $\mathcal{S}$. We show this by means of an example. With this in mind, we define an *Euclidean path* in Euclidean topological space $(X, \mathcal{C}^X)$ as any continuous function $p : (\mathbb{R}_{\geq 0}, \mathcal{C}^{\mathbb{R}_{\geq 0}}) \to (X, \mathcal{C}^X)$, where $\mathbb{R}_{\geq 0}$ is the half-line $\{x \in \mathbb{R} \mid 0 \leq x\}$ and $\mathcal{C}^{\mathbb{R}_{\geq 0}}$ is the Euclidean closure operator.

Example 8. We define two models based on the Euclidean topology over $\mathbb{R}^2$, seen as a closure space $(\mathbb{R}^2, \mathcal{C})$, where $\mathcal{C}$ is the standard closure operator in $\mathbb{R}^2$. We use propositions b, w, g, depicted in Figure 25 as black, white and grey areas, respectively. Consider the sets $H = \{(x,y)|x^2+y^2 < 1\}$, $H^{<} = \{(x,y)|x^2+y^2 = 1 \wedge x < 0\}$, $H^{\geq} = \{(x,y)|x^2+y^2 = 1 \wedge x \geq 0\}$. Let $\mathcal{M}_i = ((\mathbb{R}^2, \mathcal{C}), \mathcal{V}_i)$, for $i \in \{1,2\}$. Fix valuations as follows: $\mathcal{V}_1(b) = H \cup H^{<}$, $\mathcal{V}_1(w) = \mathbb{R}^2 \setminus \mathcal{V}_1(b)$, $\mathcal{V}_1(g) = \emptyset$, $\mathcal{V}_2(b) = \mathcal{V}_1(b)$, $\mathcal{V}_2(w) = H^{\geq} \setminus H$, $\mathcal{V}_2(g) = \mathbb{R}_2 \setminus (H \cup H^{<} \cup H^{\geq})$. Let $x \in H$. Using the path-based definition of $\mathcal{S}$ in Proposition 5, we have $\mathcal{M}_1, x \models b\,\mathcal{S}\,w$, and $\mathcal{M}_2, x \not\models b\,\mathcal{S}\,w$, as there are paths starting at a black point in $\mathcal{M}_2$ and reaching a grey point, which does not satisfy b, without passing by white points. If we consider the closure space based semantics in Definition 22, the expectation is that $b\ \mathcal{S}\ w$ holds at x in $\mathcal{M}_1$, which is true by the choice $A = H \cup H^{<}$, but note that $\mathcal{B}^+(A) = H^{\geq}$. For this reason, we also have $\mathcal{M}_2, x \models b\ \mathcal{S}\ w$ by the choice $A = H \cup H^{<}$, which is not what one would expect when thinking of the area H being "surrounded" by white points.

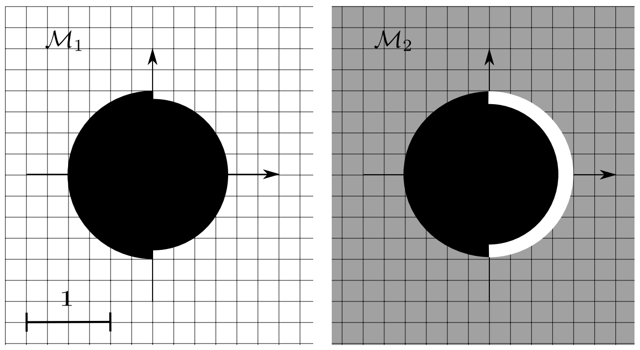


Fig. 25: Two continuous closure models (boundaries are deliberately represented as very tick, but the reader should think of them as infinitely thin).

The study of appropriate compatibility conditions, determining a universal notion of path for certain classes of closure spaces is an open issue. One of the major difficulties in finding a unifying notion is that Euclidean paths are not directed, whereas quasi-discrete paths are directed. Directed paths in topology are a highly non-trivial topic by themselves, and give rise to the subject of *directed algebraic topology* [29]. Generalising directed algebraic topology to work in the setting of closure spaces could be a relevant strategy to face these issues. We refer the interested reader to [19] for a more detailed discussion on these issues.

Topo-bisimilarity and completeness A natural question is what structures are logically equivalent, that is, how fine-grained is the logic. It turns out that in topological spaces logical equivalence for $\mathcal{L}$ (see Definition 15) coincides with the notion of *topological bisimilarity*. More precisely, fix two models $\mathcal{M}_1 = ((X_1, O_1), \mathcal{V}_1)$ and $\mathcal{M}_2 = ((X_2, O_2), \mathcal{V}_2)$.

Definition 26. *A* topological bisimulation*, or simply* topo-bisimulation*, is a relation $\mathcal{R} \subseteq X_1 \times X_2$ such that, for all $(x_1, x_2) \in \mathcal{R}$,*

- *For all $p \in AP$, $x_1 \in \mathcal{V}_1(p)$ if and only if $x_2 \in \mathcal{V}_2(p)$, and*
- *for all $o_1 \in O_1$, whenever $x_1 \in o_1$, there is $o_2 \in O_2$ such that $x_2 \in o_2$ and for all $y_2 \in o_2$ there is $y_1 \in o_1$ with $(y_1, y_2) \in \mathcal{R}$, and*
- *for all $o_2 \in O_2$, whenever $x_2 \in o_2$, there is $o_1 \in O_1$ such that $x_1 \in o_1$ and for all $y_1 \in o_1$ there is $y_2 \in o_2$ with $(y_1, y_2) \in \mathcal{R}$.*

Two points x_1, x_2 are topo-bisimilar *if there is a topo-bisimulation relating them.*

Bisimilarity equates points based on their local properties. Two points are bisimilar when, first of all, they satisfy the same properties in their respective models. Then, it is required that, for every open set o_1 on one side, there is a choice of an open set o_2 on the other side, all points of which have a corresponding bisimilar point in o_1. This distinguishes points that are on the boundary of some property from points that are in its interior. Furthermore, the precise shape and size of properties in a model does not affect bisimilarity, which is only driven by the existence of open sets covering each property.

Topo-bisimilarity establishes the same relation between points in a topological space as topo-logical equivalence, i.e. $\mathcal{L}$ is fully abstract with respect to topo-bisimilarity. This is proved in [41], Theorems 5.4 and 5.5. We summarise the result as follows.

Theorem 3. *Two points $x_1 \in X_1$ and $x_2 \in X_2$ are topo-bisimilar if and only if they are* logically equivalent*, that is, for all formulas φ, it holds $\mathcal{M}_1, x_1 \models \varphi$ if and only if $\mathcal{M}_2, x_2 \models \varphi$.*

A situation where Theorem 3 is useful is when one wants to prove that two models are logically equivalent or to minimise the model to check. Then instead of verifying equivalence over all formulas (e.g., by induction), one can exhibit a topo-bisimulation. The development of a suitable notion of bisimilarity for SLCS is part of future research. The same holds for the understanding of the axiomatic aspects of SLCS and the development of a relational semantics.

10 Acknowledgments

The authors like to thank Luca Bortolussi, Stephen Gilmore, Gianluca Grilletti, Laura Nenzi and Rytis Paškauskas who are involved in the Quanticol project and who are co-authors of the various articles on which this tutorial has been based. We like to thank Ezio Bartocci for sharing with us an earlier Matlab version of the Turing model.

References

1. Aiello, M., Pratt-Hartmann, I., van Benthem, J. (eds.): Handbook of Spatial Logics. Springer (2007)
2. Anderson, S., Bredche, N., Eiben, A.E., Kampis, G., van Steen, M.: Adaptive Collective Systems: Herding black sheep. BookSprints (2013)
3. Aydin Gol, E., Bartocci, E., Belta, C.: A formal methods approach to pattern synthesis in reaction diffusion systems. In: Proc. of CDC (2014)
4. Baier, C., Katoen, J.P.: Principles of model checking. MIT Press (2008)
5. Bartocci, E., Bortolussi, L., Milios, D., Nenzi, L., Sanguinetti, G.: Studying emergent behaviours in morphogenesis using signal spatio-temporal logic. In: HSB. Lecture Notes in Computer Science, vol. 9271, pp. 156–172. Springer (2015)
6. Bartocci, E., Bortolussi, L., Nenzi, L., Sanguinetti, G.: On the robustness of temporal properties for stochastic models. In: HSB. EPTCS, vol. 125, pp. 3–19 (2013)
7. Bartocci, E., Bortolussi, L., Nenzi, L., Sanguinetti, G.: System design of stochastic models using robustness of temporal properties. Theor. Comput. Sci. 587, 3–25 (2015)
8. Bertocci, E., Grosu, R.: Spatio-temporal model checking. In: Bernardo, M., De Nicola, R., Hillston, J. (eds.) Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2016)
9. Blackburn, P., de Rijke, M., Venema, Y.: Modal logic. Cambridge University Press, New York, NY, USA (2001)
10. Borgnat, P., Abry, P., Flandrin, P., Robardet, C., Rouquier, J.B., Fleury, E.: Shared bicycles in a city: A signal processing and data analysis perspective. Adv. Complex Syst. 14(3), 415–438 (2011)
11. Bortolussi, L., Hillston, J.: Model checking single agent behaviours by fluid approximation. Inf. Comput. 242, 183–226 (2015)
12. Bortolussi, L., Hillston, J., Latella, D., Massink, M.: Continuous approximation of collective system behaviour: A tutorial. Performance Evaluation 70(5), 317 – 349 (2013), `http://www.sciencedirect.com/science/article/pii/S0166531613000023`
13. Cardelli, L., Gordon, A.D.: Anytime, anywhere: Modal logics for mobile ambients. In: Proceedings of the 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00). pp. 365–377 (2000)
14. Ciancia, V., Gilmore, S., Grilletti, G., Latella, D., Loreti, M., Massink, M.: On spatio-temporal model-checking of vehicular movement in transport systems – preliminary version. Technical Report TR-QC-02-2016, QUANTICOL (2016)
15. Ciancia, V., Gilmore, S., Latella, D., Loreti, M., Massink, M.: Data verification for collective adaptive systems: Spatial model-checking of vehicle location data. In: Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW 2014, London, United Kingdom, September 8-12, 2014. pp. 32–37. IEEE Computer Society (2014), `http://dx.doi.org/10.1109/SASOW.2014.16`
16. Ciancia, V., Grilletti, G., Latella, D., Loreti, M., Massink, M.: An experimental spatio-temporal model checker. In: Proceedings of VERY*SCART (workshop affiliated to SEFM 2015). LNCS, Springer (2015), to appear. Extended version of QC-TR- 10-2014, http://milner.inf.ed.ac.uk/wiki/pages/J8N4c8/QUANTICOL Technical Reports.html.

17. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Specifying and verifying properties of space. Tech. Rep. TR-QC-06-2014, QUANTICOL (2014), `http://blog.inf.ed.ac.uk/quanticol/technical-reports/`
18. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Specifying and Verifying Properties of Space. In: Springer (ed.) The 8th IFIP International Conference on Theoretical Computer Science, TCS 2014, Track B. Lecture Notes in Computer Science, vol. 8705, pp. 222–235 (2014)
19. Ciancia, V., Latella, D., Loreti, M., Massink, M.: Model checking spatial logics for closure spaces (2016), submitted. Manuscript available from the authors
20. Ciancia, V., Latella, D., Massink, M., Paškauskas, R.: Exploring Spatio-temporal Properties of Bike-sharing Systems. In: Beal, J., Hillston, J., Viroli, M. (eds.) Spatial and COllective PErvasive Computing Systems. Workshop at IEEE SASO 2015, MIT, Cambridge, MA, USA, Sep. 21, 2015. pp. 74–79. IEEE Computer Society Press (2015), dOI: 10.1109/SASOW.2015.17
21. Clarke, E.M., Grumberg, O., Peled, D.: Model checking. MIT Press (2001), `http://books.google.de/books?id=Nmc4wEaLXFEC`
22. De Maio, P.: Bike-sharing: Its history, impacts, models of provision, and future. Journal of Public Transportation 12(4), 41–56 (2009)
23. De Nicola, R., Katoen, J.P., Latella, D., Loreti, M., Massink, M.: Model checking mobile stochastic logic. Theor. Comput. Sci. 382(1), 42–70 (2007)
24. Fishman, E., Washington, S., Haworth, N.L.: Bike share's impact on car use: evidence from the United States, Great Britain, and Australia. In: Proceedings of the 93rd Annual Meeting of the Transportation Research Board (2014)
25. Froehlich, J., Neumann, J., Oliver, N.: Sensing and predicting the pulse of the city through shared bicycling. In: IJCAI. pp. 1420–1426 (2009)
26. Galpin, V.: Spatial representations and analysis techniques. In: Bernardo, M., De Nicola, R., Hillston, J. (eds.) Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2016)
27. Galton, A.: A generalized topological view of motion in discrete space. Theoretical Computer Science 305(1–3), 111 – 134 (2003), `http://www.sciencedirect.com/science/article/pii/S0304397502007016`
28. Galton, G.: The mereotopology of discrete space. In: Freksa, C., Mark, D. (eds.) Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science. Lecture Notes in Computer Science, vol. 1661, pp. 251–266. Springer Berlin Heidelberg (1999), `http://dx.doi.org/10.1007/3-540-48384-5_17`
29. Grandis, M.: Directed algebraic topology : models of non-reversible worlds. Cambridge University Press, Cambridge (2009)
30. Haghighi, I., Jones, A., Kong, J.Z., Bartocci, E., R., G., Belta, C.: SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems. In: Proc. of HSCC (2015)
31. Johnstone, P.T.: Sketches of an elephant : a topos theory compendium. Vol. 1. Oxford Logic Guides, Clarendon Press, Oxford (2002), `http://opac.inria.fr/record=b1107183`, autre tirage : 2008
32. Kontchakov, R., Kurucz, A., Wolter, F., Zakharyaschev, M.: Spatial logic + temporal logic = ? In: Aiello et al. [1], pp. 497–564
33. Latella, D., Loreti, M., Massink, M.: On-the-fly PCTL fast mean-field approximated model-checking for self-organising coordination. Sci. Comput. Program. 110, 23–50 (2015)

34. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Proc. FORMATS (2004)
35. Massink, M., Paškauskas, R.: Model-based Assessment of Aspects of User-satisfaction in Bicycle Sharing Systems. In: Sotelo Vazquez, M., Olaverri Monreal, C., Miller, J., Broggi, A. (eds.) 18th IEEE International Conference on Intelligent Transportation Systems. pp. 1363–1370. IEEE, IEEE Computer Society Press (2015), dOI: 10.1109/ITSC.2015.224
36. Midgley, P.: Bicycle-sharing schemes: Enhancing sustainable mobility in urban areas. In: 19th session of the Commission on Sustainable Development. CSD19/2011/BP8, United Nations (2011)
37. Nenzi, L., Bortolussi, L.: Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic. In: Haviv, M., Knottenbelt, W.J., Maggi, L., Miorandi, D. (eds.) 8th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2014, Bratislava, Slovakia, December 9-11, 2014. ICST (2014), http://dx.doi.org/10.4108/icst.valuetools.2014.258183
38. Nenzi, L., Bortolussi, L., Ciancia, V., Loreti, M., Massink, M.: Qualitative and quantitative monitoring of spatio-temporal properties. In: Bartocci, E., Majumdar, R. (eds.) Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings. Lecture Notes in Computer Science, vol. 9333, pp. 21–37. Springer (2015), http://dx.doi.org/10.1007/978-3-319-23820-3_2
39. Reynolds, J.: Separation logic: A logic for shared mutable data structures. In: 17th IEEE Symposium on Logic in Computer Science (LICS 2002), 22-25 July 2002, Copenhagen, Denmark, Proceedings. pp. 55–74. IEEE Computer Society (2002), http://dx.doi.org/10.1109/LICS.2002.1029817
40. Turing, A.M.: The Chemical Basis of Morphogenesis. Philosophical Transactions of the Royal Society of London B: Biological Sciences (1952)
41. van Benthem, J., Bezhanishvili, G.: Modal logics of space. In: Handbook of Spatial Logics, pp. 217–298. Springer (2007)