Logical Methods in Computer Science
Vol. 12(4:2)2016, pp. 1–51
www.lmcs-online.org

Submitted Feb. 8, 2016
Published Oct. 11, 2016

MODEL CHECKING SPATIAL LOGICS FOR CLOSURE SPACES *

VINCENZO CIANCIA [a], DIEGO LATELLA [b], MICHELE LORETI [c], AND MIEKE MASSINK [d]

[a,b,d] Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo" - CNR, Pisa
e-mail address: {vincenzo.ciancia, diego.latella, mieke.massink}@isti.cnr.it

[c] Università degli Studi di Firenze and IMT Alti Studi, Lucca
e-mail address: michele.loreti@unifi.it

Abstract. Spatial aspects of computation are becoming increasingly relevant in Computer Science, especially in the field of *collective adaptive systems* and when dealing with systems distributed in physical space. Traditional formal verification techniques are well suited to analyse the temporal evolution of programs; however, properties of space are typically not taken into account explicitly. We present a topology-based approach to formal verification of spatial properties depending upon physical space. We define an appropriate logic, stemming from the tradition of topological interpretations of modal logics, dating back to earlier logicians such as Tarski, where modalities describe neighbourhood. We lift the topological definitions to the more general setting of *closure spaces*, also encompassing discrete, graph-based structures. We extend the framework with a spatial *surrounded* operator, a *propagation* operator and with some *collective* operators. The latter are interpreted over arbitrary *sets* of points instead of individual points in space. We define efficient model checking procedures, both for the individual and the collective spatial fragments of the logic and provide a proof-of-concept tool.

1. Introduction

Much attention has been devoted in Computer Science to formal verification of process behaviour. Several techniques have been studied and developed that are based on a formal understanding of system requirements through *modal* logics. Such logics typically have a *temporal* flavour, describing the flow of events, and are interpreted in various kinds of transition structures. Among those techniques *model checking* is one of the most successful (for an extensive overview see e.g. [BK08] and references therein).

In recent times, aspects of computation related to the distribution of systems in physical space have become increasingly relevant. An example is provided by so called *collective adaptive systems*[1]. Such systems are typically composed of a large number of interacting

2012 ACM CCS: [**Theory of computation**]: Logic—Logic and verification / Modal and temporal logics / Verification by model checking; Semantics and reasoning; [**Software and its engineering**]: Software organization and properties—Software functional properties—Formal methods—Software verification;

Key words and phrases: Spatial Logics, Spatial Model Checking, Closure Spaces, Collective Logics.

* Research partially funded by EU project QUANTICOL (nr. 600708).

[1] See e.g. the web site of the QUANTICOL project: `http://www.quanticol.eu`, and that of the FOCAS Coordination Action: `http://www.focas.eu`.

LOGICAL METHODS IN COMPUTER SCIENCE
DOI:10.2168/LMCS-12(4:2)2016

objects located in space. Their global behaviour critically depends on interactions which are often local in nature. The aspect of locality immediately poses issues of spatial distribution of objects. Abstraction from spatial distribution may sometimes provide insights in the system behaviour, but this is not always the case. For example, consider a bike (or car) sharing system having several parking stations, and featuring twice as many parking slots as there are vehicles in the system. Ignoring the spatial dimension, on average, the probability to find completely full or empty parking stations at an arbitrary station is very low; however, this kind of analysis may be misleading, as in practice some stations are much more popular than others, often depending on nearby points of interest. This leads to quite different probabilities to find stations completely full or empty, depending on the examined location. In other cases, it may be important to be able to specify spatial properties concerning *groups* of points in space rather than of individual points. For example, the property that agents associated to points in space are able to connect to one another and act as a group, or that they are located all together in a protected environment, or that they can share part of the same route to reach a common exit or goal. In all such situations, it is important to be able to predicate over spatial aspects, and eventually find methods to certify that a given collective adaptive system satisfies specific requirements in this respect.

In Logics, there is a considerable amount of literature focused on so called *spatial* logics, that is, a spatial interpretation of modal logics [APHvB07]. Dating back to early logicians such as Tarski, modalities may be interpreted using the concept of *neighbourhood* in a topological space. The field of spatial logics is well developed in terms of descriptive languages and decidability or complexity aspects. However, in this field, scant attention has been devoted to date to the development of formal and automatic verification methods, e.g. model checking. Furthermore, the formal treatment of *discrete* models of space is still a relatively unexplored field, with notable exceptions such as the work by Rosenfeld [KR89, Ros79], Galton (e.g.[Gal14, Gal03, Gal99]) and by Smyth and Webster [SW07]. Kovalevsky [Kov08] studied alternative axioms for topological spaces in order to recover well-behaved notions of neighbourhood. The outcome is that one may impose closure operators on top of a topology, that do not coincide with topological closure.

In [CLLM14a] we proposed the logic SLCS (Spatial Logic for Closure Spaces), extending the topological semantics of modal logics to *closure spaces*. The work follows up on the research line of Galton and Smyth and Webster, enhancing it with a modal logic perspective. Closure spaces (also called *Čech closure spaces* or *preclosure spaces* in the literature) are based on a single operator on sets of points, namely the *closure* operator, and are a generalisation of standard topological spaces. In addition, finite spaces and graphs are subclasses of closure spaces and the graph-theoretical notion of neighbourhood coincides with the notion of neighbourhood defined in the context of closure spaces. Thus, closure spaces provide a uniform framework for the treatment of all major models of space.

We provided a logical operator corresponding to the closure operator on sets of points in space, and a spatial interpretation of the temporal *until* operator, fundamental in the classical temporal setting, arriving at the definition of a logic which is able to describe unbounded areas of space. Intuitively, the spatial until operator, which in the present paper we call *surrounded*, describes a situation in which it is not possible to "escape" an area of points satisfying a certain property, unless by passing through at least one point that satisfies another given formula. This operator is similar in spirit to the spatial until operator for topological spaces discussed by Aiello and van Benthem in [Aie02, BB07]. In [CLLM14a] we also presented a model-checking algorithm for SLCS when interpreted on finite models.

The combination of SLCS with temporal operators from the well-known branching time logic CTL (Computation Tree Logic) [CE82], has been explored in [CGL$^+$15, CLMP15] and provides spatio-temporal reasoning and model checking.

In the present paper we extend SLCS with a further operator, $\mathcal{P}$, capturing the notion of spatial propagation; intuitively the formula $\phi\ \mathcal{P}\ \psi$ describes a situation in which the points satisfying ψ can be reached by paths rooted in points satisfying ϕ and, for the rest, composed only of points satisfying ψ. We furthermore extend the logic with operators for *collective properties*, namely properties which are satisfied by *connected sets* of points, rather than points in isolation. The formal semantics of the extended logic—CSLCS, Collective SLCS–are provided in the form of a satisfiability relation defined using the notion of infinite path in closure spaces. We finally extend the model-checking algorithm in order to treat the newly introduced operators, and we present several examples of use of SLCS and CSLCS from the domain of collective adaptive systems using a prototype implementation of the spatial model-checker.

Related work. Variants of spatial logics have also been proposed for the symbolic representation of the contents of images, and, combined with temporal logics, for sequences of images [DBVZ95]. The latter approach is based on a discretisation of the space of the images in rectangular regions and the orthogonal projection of objects and regions onto Cartesian coordinate axes such that their possible intersections can be analysed from different perspectives. It involves two spatial until operators defined on such projections considering spatial shifts of regions along the positive, respectively negative, direction of the coordinate axes and it is very different from the topological spatial logic approach.

In [GBC$^+$08, GSC$^+$09, GBB14] another variant of spatial logic is proposed in which spatial properties are expressed using ideas from image processing, namely quad trees. This variant is equipped with practical model checking algorithms and with machine learning procedures and allows one to capture very complex spatial structures. However, this comes at the price of a complex formulation of spatial properties, which need to be learned from some template image. The combination of this spatial logic with linear time signal temporal logic, defined with respect to continuous-valued signals, has recently led to the spatio-temporal logic SpaTeL [HJK$^+$15].

In the specific setting of complex and collective adaptive systems, techniques for efficient approximation have been developed in the form of mean-field or fluid-flow analysis (see [BHLM13] for a tutorial introduction). Recently (see for example [CLBR09]), the importance of spatial aspects has been recognised and studied in this context. In [NB14] a first step towards the combination of signal temporal logic with spatial operators such as 'somewhere' and 'everywhere' has been performed. These two operators were also proposed in work by Reif and Sistla [RS85]. In further joint work along these lines [NBC$^+$15] some of the spatial operators based on closure spaces from SLCS, such as the 'surrounded' operator, have been added to the signal temporal logic fraction. Both boolean semantics and quantitative semantics of the spatio-temporal logic have been provided. The quantitative semantics provide a measure of the robustness with which a spatio-temporal property holds in a given point in space at a particular time. The approach has been applied to investigate the emergence and persistence of Turing patterns in animal fur based on reaction diffusion models.

In [CG12] a geometric process algebra based on affine geometry has been proposed for describing the concurrent evolution of geometric structures in 3D space. Spatial dynamics of

systems have also been studied in the context of Systems Biology applying suitable modelling and simulation approaches. In [JEU08] a spatial (and temporal) extension of the π-Calculus is proposed. The notion of space is expressed by associating each process with its current position in $\mathbb{R}^d$. The formal semantics of the language is given, based on which simulation tools have been developed. In [BHMU11] an attributed, multi-level, rule-based language, ML-Space, is presented that allows one to integrate different types of spatial dynamics within one model. The associated simulator combines several stochastic simulation methods. This allows for the simulation of reaction diffusion systems as well as taking excluded volume effects into account. Formal verification and analysis, e.g. model checking, is not addressed.

In the Computer Science literature, some spatial logics have been proposed, that typically describe situations in which modal operators are interpreted *syntactically* against the structure of agents in a process calculus. We refer to [CG00, CC03] for some classical examples. In the same line, a recent example is given by [TPGN15], concerning model checking of security aspects in cyber-physical systems, in a spatial context based on the idea of bigraphical reactive systems introduced by Milner [Mil09]. The objects of discussion in the latter research lines are operators that for example quantify over the parallel sub-components of a system, the containment relation between places, or the hidden resources of an agent. The meaning of the terminology "spatial logics" in that case is different from that used in the present paper, where the "topological" interpretation of [BB07] is intended. The influence of space on agents interaction is also considered in the literature on process calculi using *named locations* [DFP98], where every process interaction primitive is enriched with the indication of (the name of) the location where the action operates. In that paper, space is modelled as a discrete, finite set of points.

Logics for graphs have been studied in the context of databases and process calculi (see [CGG02, GL07], and the references therein), even though the relationship with physical space is often only implicit, if considered at all.

Graph-based spatial logics for collective adaptive systems are also proposed in [AS15]. In that approach the logic extends a chemical-based coordination model based on logic inference. Properties are expressed in the form of combinations of logic programs. The spatial operators distribute such programs over the nodes of a graph to infer information local to each node. The locally inferred data is logically aggregated at local spatial locations. Evaluated properties involve collective aspects either with a local scope (neighbourhood) or with a global scope. The approach relies on *a priori* defined spatial patterns.

A successful attempt to bring topology and digital imaging together is represented by the field of *digital topology* [Ros79, KR89]. In spite of its name, this area studies digital images using models inspired by topological spaces, but neither generalising nor specialising these structures. Rather recently, closure spaces have been proposed as an alternative foundation of digital imaging by various authors, especially Smyth and Webster [SW07] and Galton [Gal03]; we continue that research line in the present paper, enhancing it with a (modal) logic perspective.

In [Gal14], a sub-class of closure spaces, namely *adjacency spaces*, is presented. An adjacency space is characterized by a set of entities together with a reflexive and symmetric relation. In the above mentioned paper, adjacency spaces are used as the basis for the definition of *regions*, i.e. *sets of* entities, and the construction of a discrete interpretation of logical operators typical of region calculi, based on the notion of region *connectedness* derived from the notion of entity adjacency. Region calculi operators predicate on regions (see [KKWZ07] for a comprehensive overview), using boolean connectives like "part of",

"boundary", "overlap" and so on. An important aspect of adjacency spaces is that they can be easily turned into topological spaces, without loosing any information on their internal structure, which makes them rather attractive. Verification issues, e.g. model-checking, are not addressed in [Gal14].

The structure of the present paper is as follows. Section 2 recalls basic concepts and definitions related to closure spaces, their sub-classes of topological spaces and quasi-discrete closure spaces and introduces the notion of Euclidean and quasi-discrete paths in closure spaces. Section 3 briefly recalls SLCS and presents its extension with the *propagation* operator $\mathcal{P}$. Section 4 introduces the collective spatial logic CSLCS while Section 5 shows some examples of use of the proposed logics when interpreted on quasi-discrete closure spaces. In Section 6 the model-checking algorithms for SLCS and CSLCS interpreted on finite models are presented. In Section 7 the proof-of-concept model-checker is shown together with several examples of use. Finally, some conclusions are drawn and lines for future research are outlined in Section 8. All detailed proofs are provided in the Appendix.

2. Topological and Closure spaces

In this work, we resort to some abstract mathematical structures for the definition of space. The mathematical structure of choice of spatial logics are very often *topological spaces*, possibly enriched with metrics, or other spatial features (see [BB07]). The use of abstract structures has the advantage to separate logical operators, such as neighbourhood, from the specific nature of space (e.g., the number of dimensions, or the presence or absence of metric features, etc.). However, using topological spaces, it may be difficult to deal with discrete structures, such as finite graphs. In [Gal03], *closure spaces*, which generalise topological spaces, are proposed as a unifying approach treating both topological spaces and graphs in a satisfactory way. In this section, we recall several definitions and results on topological and closure spaces, most of which are taken from [Gal03].

2.1. Topological spaces.

We will first provide the basic definitions that are used to relate closure spaces to the more widely known topological spaces. The link between topological and closure spaces is deep. In this section we provide a brief introduction to the topic; we refer the reader to, e.g., [Gal03] for more information.

Definition 2.1. 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.

Definition 2.2. In a topological space (X, O), $A \subseteq X$ is *closed* if its complement is open.

Definition 2.3. In a topological space (X, O), the *closure* of $A \subseteq X$ is the *least* closed set containing A.

We remark that closure is well-defined as arbitrary intersections of closed sets are closed, and X itself is both open and closed. An alternative, equivalent formulation of topological spaces is given by the Kuratowski definition.

Definition 2.4. According to the Kuratowski definition, a topological space is a pair $(X, \mathcal{C})$ where X is a set, and the *closure operator* $\mathcal{C} : \wp(X) \to \wp(X)$ assigns to each subset of X its *closure*, obeying to the following laws, for all $A, B \subseteq X$:

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

The Kuratowski and open sets definitions of a topological space are equivalent. The proof can be sketched as follows. To obtain the Kuratowski definition from a topological space defined in terms of open sets, one defines $\mathcal{C}(A)$ as topological closure (Definition 2.3). The properties of Definition 2.4 can be shown to hold. For the converse, starting from a Kuratowski topological space $(X, \mathcal{C})$, the open sets are defined as those sets A that are equal to their *interior*, that is, $A = \overline{\mathcal{C}(\overline{A})}$ where for any $B \subseteq X$ we let $\overline{B}$ denote the *complement* of B, i.e. $X \setminus B$.

2.2. **Closure spaces.** A *closure space* (also called *Čech closure space* or *preclosure space* in the literature), is composed of a set (of points) and a (closure) operator on subsets (of points), as specified by the following definition:

Definition 2.5. A *closure space* is a pair $(X, \mathcal{C})$ where X is a set, and the *closure operator* $\mathcal{C} : \wp(X) \to \wp(X)$ assigns to each subset of X its *closure*, obeying to the following laws, for all $A, B \subseteq X$:
(1) $\mathcal{C}(\emptyset) = \emptyset$;
(2) $A \subseteq \mathcal{C}(A)$;
(3) $\mathcal{C}(A \cup B) = \mathcal{C}(A) \cup \mathcal{C}(B)$.

Closure spaces are a generalisation of *topological spaces*, which is easy to see by comparing Definition 2.5 with Definition 2.4; the difference is that the *idempotency axiom* $\mathcal{C}(\mathcal{C}(A)) = \mathcal{C}(A)$ is not required in closure spaces. Indeed, topological spaces are precisely the subclass of closure spaces where such axiom holds. We shall call a closure space *topological* or *idempotent* or *Kuratowski* in that case. We note in passing that the notion of *continuous function* also extends to closure spaces (see Definition 2.30), making closure spaces a *category* in the sense of category theory, and topological spaces a *full subcategory*.

Below, we consider an example of a closure space, with set of points X in a classical Euclidean space, but exhibiting a non-standard closure operator.

Example 2.6. Let $\delta \in \mathbb{R}_{>0}$ and $\mathcal{C}_\delta : \wp(\mathbb{R}^2) \to \wp(\mathbb{R}^2)$ be such that:

$$\mathcal{C}_\delta(A) = \{(x_1, y_1) \in \mathbb{R}^2 | \exists (x_2, y_2) \in A. \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \leq \delta\}$$

Function $\mathcal{C}_\delta$ maps each subset A of $\mathbb{R}^2$ to the set of points located in a radius δ from a point in A (see Figure 1). It is easy to see that $\mathcal{C}_\delta$ satisfies all the three conditions of Definition 2.5 and that $(\mathbb{R}^2, \mathcal{C}_\delta)$ is a closure space.

Example 2.7. The closure space of Example 2.6 is *not* a *topological* space, as its closure operator is not idempotent.

Definition 2.8. Let $(X, \mathcal{C})$ be a closure space; for each $A \subseteq X$:
(1) the *interior* $\mathcal{I}(A)$ of A is the set $\overline{\mathcal{C}(\overline{A})}$;
(2) A is a *neighbourhood* of $x \in X$ if and only if $x \in \mathcal{I}(A)$;
(3) A is *closed* if $A = \mathcal{C}(A)$ while it is *open* if $A = \mathcal{I}(A)$.

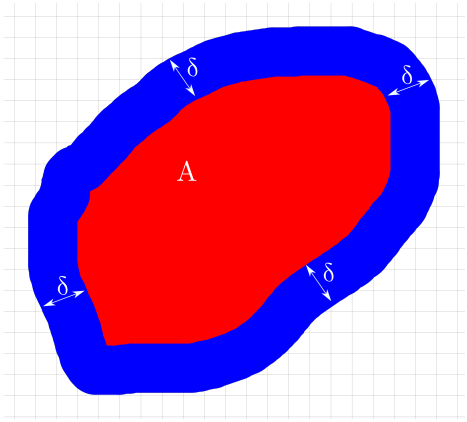

Figure 1: A picture of Example 2.6; the union of the blue and red areas is the closure of the red area

Example 2.9. Let us consider the closure space $(\mathbb{R}^2, \mathcal{C}_\delta)$, introduced in Example 2.6, assuming, for simplicity, that $\delta \leq 1$. Let $A = \{(x,y) \in \mathbb{R}^2 | \sqrt{x^2+y^2} \leq 1\}$. We have that:

- $\mathcal{I}(A) = \{(x,y) \in \mathbb{R}^2 | \sqrt{x^2+y^2} \leq 1-\delta\}$;
- for any $(x_1, y_1) \in \mathbb{R}^2$, A is a neighbourhood of (x_1, y_1) if and only if:
$$\{(x_2,y_2) \in \mathbb{R}^2 | \sqrt{(x_2-x_1)^2+(y_2-y_1)^2} \leq \delta\} \subseteq A$$
- the only *closed* set (of the closure operator $\mathcal{C}_\delta$) in $\wp(\mathbb{R}^2)$ is $\mathbb{R}^2$, while $\emptyset$ is the only *open* set.

The following proposition states a number of general properties of closure spaces.

Proposition 2.10. *Let $(X, \mathcal{C})$ be a closure space, the following properties hold:*

(1) *$A \subseteq X$ is open if and only if $\overline{A}$ is closed;*
(2) *closure and interior are monotone operators over the inclusion order, that is: $A \subseteq B \implies \mathcal{C}(A) \subseteq \mathcal{C}(B)$ and $\mathcal{I}(A) \subseteq \mathcal{I}(B)$*
(3) *Finite intersections and arbitrary unions of open sets are open.*

Given a closure space $(X, \mathcal{C})$, and $A \subseteq X$, we can define the *boundary* of A. The latter is only given in terms of closure and interior, and coincides with the definition of boundary in a topological space. We also provide two similar notions, namely the *interior* and *closure* boundary (the latter is sometimes called *frontier*).

Definition 2.11. 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 $\mathcal{B}^+(A) = \mathcal{C}(A) \setminus A$.

In [Gal99], 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 from a reflexive and symmetric relation (see Definition 2.16 in the next section). Therein, in Lemma 5, it is proved that the definition of [Gal99] coincides with the one we provide above.

Proposition 2.12. *The following equations hold in a closure space:*

$$\mathcal{B}(A) = \mathcal{B}^+(A) \cup \mathcal{B}^-(A) \tag{2.1}$$
$$\mathcal{B}^+(A) \cap \mathcal{B}^-(A) = \emptyset \tag{2.2}$$
$$\mathcal{B}(A) = \mathcal{B}(\overline{A}) \tag{2.3}$$
$$\mathcal{B}^+(A) = \mathcal{B}^-(\overline{A}) \tag{2.4}$$
$$\mathcal{B}^+(A) = \mathcal{B}(A) \cap \overline{A} \tag{2.5}$$
$$\mathcal{B}^-(A) = \mathcal{B}(A) \cap A \tag{2.6}$$
$$\mathcal{B}(A) = \mathcal{C}(A) \cap \mathcal{C}(\overline{A}) \tag{2.7}$$

A closure space can be also obtained by restricting the domain of another space.

Definition 2.13. Given a closure space $(X, \mathcal{C})$ and a subset $Y \subseteq X$, we call *subspace closure* the operation $\mathcal{C}^Y : \wp(Y) \to \wp(Y)$ defined as $\mathcal{C}^Y(A) = \mathcal{C}(A) \cap Y$. We call $(Y, \mathcal{C}^Y)$ the subspace of $(X, \mathcal{C})$ *generated by* Y.

Proposition 2.14. *The subspace closure is a closure operator.*

Example 2.15. $(\mathbb{R}^2_{\geq 0}, \mathcal{C}_\delta^{\mathbb{R}^2_{\geq 0}})$ is a subspace of the closure space $(\mathbb{R}^2, \mathcal{C}_\delta)$ introduced in Example 2.6, generated by $\mathbb{R}^2_{\geq 0}$.

2.3. **Quasi-discrete closure spaces.** A closure space may be derived starting from a *binary relation*, that is, a *graph*. Such closure spaces may be characterised as *quasi-discrete* as briefly presented in this section. For additional details we refer the interested reader to [Gal03].

Definition 2.16. Consider a set X and a relation $R \subseteq X \times X$. A closure operator is obtained from R as $\mathcal{C}_R(A) = A \cup \{x \in X \mid \exists a \in A.(a, x) \in R\}$.

Proposition 2.17. *The pair $(X, \mathcal{C}_R)$ is a closure space.*

Closure operators obtained by Definition 2.16 are not necessarily idempotent. Lemma 11 in [Gal03] provides a necessary and sufficient condition, that we rephrase below. We let $R^=$ denote the reflexive closure of R, that is, the smallest reflexive relation containing R, which is defined as the union of R with the identity relation on the same domain.

Lemma 2.18. *$\mathcal{C}_R$ is idempotent if and only if $R^=$ is transitive.*

Note that when R is transitive, so is $R^=$, thus $\mathcal{C}_R$ is idempotent. The vice-versa is not true. For instance, it may happen that $(x, y) \in R$, and $(y, x) \in R$, but $(x, x) \notin R$.

Remark 2.19. In topology, open sets play a fundamental role. However, the situation is different in closure spaces derived from a relation R. For example, in a closure space derived from a symmetric relation, whose graph is connected, the only open sets are the whole space, and the empty set.

Proposition 2.20. *Given $R \subseteq X \times X$, in the space $(X, \mathcal{C}_R)$, we have:*

$$\mathcal{I}(A) = \{x \in A \mid \neg\exists a \in \overline{A}.(a, x) \in R\} \tag{2.8}$$
$$\mathcal{B}^-(A) = \{x \in A \mid \exists a \in \overline{A}.(a, x) \in R\} \tag{2.9}$$
$$\mathcal{B}^+(A) = \{x \in \overline{A} \mid \exists a \in A.(a, x) \in R\} \tag{2.10}$$

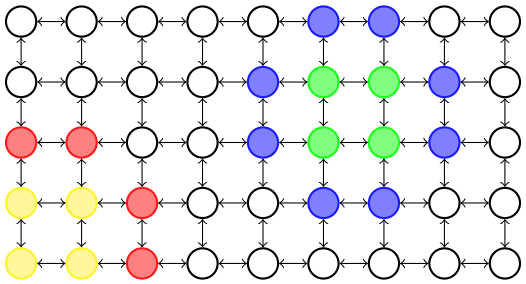

Figure 2: A graph inducing a *quasi-discrete* closure space

Closure spaces derived from a relation can be characterised as *quasi-discrete* spaces (see also Lemma 9 of [Gal03] and the subsequent statements).

Definition 2.21. 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*[2] 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 [Gal03].

Theorem 2.22. *A closure space $(X, \mathcal{C})$ is quasi-discrete if and only if there is a relation $R \subseteq X \times X$ such that $\mathcal{C} = \mathcal{C}_R$.*

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 true in a closure space when there is some R such that the closure operator of the space is derived from R. In the remainder of this section, we exemplify some aspects of quasi-discreteness.

Example 2.23. Every graph induces a *quasi-discrete* closure space. For instance, 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* nodes and *red* nodes, while the closure $\mathcal{C}_R(G)$ contains all *green* nodes and *blue* nodes. The interior $\mathcal{I}(Y)$ of Y contains a single node, the one located at the bottom-left in Figure 2. The *interior* $\mathcal{I}(G)$ of G is empty. Indeed, we have that $\mathcal{B}(G) = \mathcal{C}_R(G)$, while $\mathcal{B}^-(G) = G$ and $\mathcal{B}^+(G)$ consists of the *blue* nodes.

Example 2.24. The closure space of Example 2.6 is a quasi discrete closure space. Indeed, define $R_\delta \subseteq \mathbb{R}^2 \times \mathbb{R}^2$ as:

$$R_\delta = \{((x_1, y_1), (x_2, y_2)) | \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \leq \delta\}$$

It is easy to prove that $\mathcal{C}_\delta = \mathcal{C}_{R_\delta}$. Note that R_δ is reflexive but not transitive. So, the closure space is not a topological space.

Existence of minimal neighbourhoods does not depend on finiteness of the space; moreover, it is not even required that each point has a finite neighbourhood, as illustrated by the following example:

[2] A *minimal neighbourhood* of x is a set that is a neighbourhood of x (Definition 2.8 (2)) and is included in all other neighbourhoods of x.

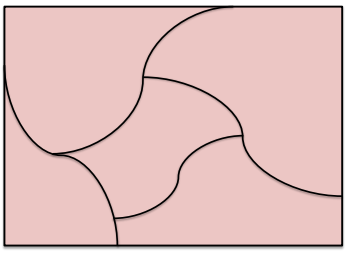

Figure 3: A quasi-discrete closure space inducing a spatial structure.

Example 2.25. Consider the rational numbers $\mathbb{Q}$, 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* (but not finite). For any $x \in \mathbb{Q}$, we have $N_x = \{y \in \mathbb{Q} | y \leq x\}$, which is not finite.

Example 2.26. Another example of closure space exhibiting minimal neighbourhoods in absence of finite neighbourhoods is the one considered in Example 2.6. In Example 2.9 we show that for any $(x_1, y_1) \in \mathbb{R}^2$, A is a neighbourhood of (x_1, y_1) if and only if

$$\{(x_2, y_2) \in \mathbb{R}^2 | \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \leq \delta\} \subseteq A.$$

Hence, $N_{(x_1,y_1)} = \{(x_2, y_2) \in \mathbb{R}^2 | \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \leq \delta\}$.

Example 2.27. 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 2.21. 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 $\bigcup_{z \in (x,y)} \mathcal{C}(z) = \bigcup_{z \in (x,y)} \{z\} = (x, y) \neq [x, y]$.

We note in passing that *any* finite space is trivially a quasi-discrete closure space. Quasi discrete closure spaces can be used to model spatial structures in $\mathbb{R}^n$, as shown below.

Example 2.28. Let $\mathscr{F} \subseteq \wp(\mathbb{R}^n)$ be a partition of $\mathbb{R}^n$, each element of which is either *open* or *closed*, i.e. $\cup_{A \in \mathscr{F}} A = \mathbb{R}^n$, $\forall A, B \in \mathscr{F} : A \neq B \rightarrow A \cap B = \emptyset$, $\forall A \in \mathscr{F} : A = \mathcal{I}(A) \vee A = \mathcal{C}(A)$. We let $R^{\mathscr{F}} \subseteq \mathscr{F} \times \mathscr{F}$ be the *connectedness* relation among elements of $\mathscr{F}$, formally:

$$R^{\mathscr{F}} = \{(A, B) | A, B \text{ open and } \mathcal{C}(A) \cap \mathcal{C}(B) \neq \emptyset\}$$

where $\mathcal{C}$ is the standard topological closure over $\mathbb{R}^n$. It is easy to see that $(\mathscr{F}, \mathcal{C}_{R^{\mathscr{F}}})$ is a quasi discrete closure space. Figure 3 shows an example in $\mathbb{R}^2$, where the open sets are shown in pink, while the only closed set is shown in black.

In Figure 4, the hierarchy of closure spaces with respect to quasi-discreteness is shown. All finite spaces are quasi-discrete, as closure of arbitrary sets is determined by that of the singletons, by the axiom $\mathcal{C}(A) \cup \mathcal{C}(B) = \mathcal{C}(A \cup B)$. 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 to any complete graph. In this case, for any set, $\mathcal{C}(A)$ is the whole space, thus closure is idempotent. More precisely, the topology determined by the closure operator associated to a complete graph is the *indiscrete* topology, where the only open sets are the empty set and the whole space. It is obvious that there are topological spaces that are not quasi-discrete, such as Euclidean spaces. Finally there are closure spaces that are neither topological nor quasi discrete. The most obvious example is

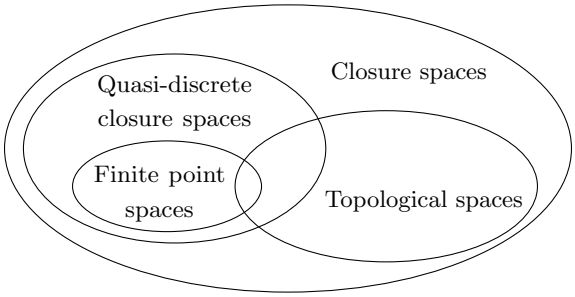

Figure 4: The hierarchy of closure spaces.

the *coproduct* (disjoint union) of a topological space which is not quasi-discrete (e.g. any Euclidean space), and a quasi-discrete, but not topological, closure space. The disjoint union of two closure spaces is defined below (we omit the proof that it actually obeys to the axioms of a closure space, as it is an easy exercise).

Definition 2.29. 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 $\mathcal{C}(A) = \{(1, x) \mid x \in \mathcal{C}^X(A^X)\} \cup \{(2, y) \mid y \in \mathcal{C}^Y(A^Y)\}$.

2.4. **Paths and connectedness in closure spaces.** In this section we define paths and connectedness for interesting classes of closure spaces. A uniform definition of paths in closure spaces is non-trivial. It is possible, and often done, to borrow the notion of path from topology. However, as we shall see, 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. Our solution is pragmatic. We define paths as it is natural in interesting classes of closure spaces. 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 truly uniform notion of path (e.g., by some form of category-theoretical *universal property* characterising a path-connected class of spaces) is left for future work. First of all we introduce the definition of continuous function, which restricts to topological continuity in the setting of idempotent closure spaces[3].

Definition 2.30. 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 $f(\mathcal{C}_1(A)) \subseteq \mathcal{C}_2(f(A))$.

Below, two kinds of paths are introduced: *Euclidean* paths and *quasi-discrete* paths.

Definition 2.31. For each closure space $(X, \mathcal{C})$, assume a chosen closure space $\mathfrak{I}$, equipped with a linear order $\leq$ with bottom 0, and call path a continuous function $p : \mathfrak{I} \to (X, \mathcal{C})$. In particular, call *Euclidean path* any continuous function whose domain is the half-line $\mathbb{R}_{\geq 0} = \{x \in \mathbb{R} \mid 0 \leq x\}$, equipped with the Euclidean (topological) closure operator. Call

[3]Note that in topological spaces one may equivalently use the definition we propose here, based on the Kuratowski axioms, or the definition of continuity using open sets, namely f is continuous whenever for each open set o, $f^{-1}(o)$ is open. However, the two definitions do not coincide for arbitrary closure spaces (open sets play a less important role in closure spaces, see Remark 2.19).

quasi-discrete path any continuous function whose domain is the quasi-discrete closure space $(\mathbb{N}, \mathcal{C}_{Succ})$ where $(n, m) \in Succ \iff m = n+1$. Whenever $(X, \mathcal{C})$ is an Euclidean topological space (resp. a quasi-discrete closure space), call *path* an Euclidean (resp. quasi-discrete) path whose codomain is $(X, \mathcal{C})$.

Note that in Definition 2.31 we do not require compatibility conditions between the closure operator and the linear order of $\mathfrak{I}$. Depending on the application context, different orders may be chosen, obtaining different interpretations of logics, or different degrees of compatibility between closure and paths (see e.g. Theorem 3.7). We consider the study of appropriate compatibility conditions, determining a universal notion of path for certain classes of closure spaces, out of scope for the current paper. We can, though, provide a hint about the complexity of such study. One of the major difficulties in finding a unifying notion is that Euclidean paths are not directed, whereas quasi-discrete paths are directed. The examples in this section are also aimed at making this problem more clear. Directed paths in topology are a highly non-trivial topic by themselves, and gave rise to the subject of *directed algebraic topology* [Gra09]. Generalizing directed algebraic topology to work in the setting of closure spaces could be a relevant strategy to face these issues.

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 $p(i) = y$. We also write $p : x \overset{i}{\underset{y}{\rightsquigarrow}} \infty$ when p is a path from x and $p(i) = y$.

The definition of Euclidean path is intuitively similar to the classical topological definition of a path, namely a continuous function from the unit interval $[0, 1]$, except that Euclidean paths that we defined are "open-ended on the right" (note that the open interval $[0, 1)$ and $\mathbb{R}^+$ are continuously isomorphic). The definition of quasi-discrete path, on the other hand, mimics the classical definition of infinite path in a graph. Simply adopting Euclidean paths in quasi-discrete spaces yields counter-intuitive results, as shown below.

Example 2.32. Consider the quasi-discrete closure space obtained from the graph $G = (\{a, b\}, \{(b, a)\})$ having two nodes a, b, and only one edge, from b to a. Note that there is no graph-theoretical path from a to b. However, consider the function $p : \mathbb{R}_{\geq 0} \to \{a, b\}$, defined by $p(0) = a$, and $p(i) = b$ for $i \neq 0$. This function is continuous, thus it is an Euclidean path starting from a and traversing b. To see this, choose any subset J of the half-line.

- If $J = \emptyset$, the thesis is trivially obtained; otherwise, assuming $J \neq \emptyset$:
- if $J = \{0\}$, then $p(\mathcal{C}(J)) = p(\{0\}) = p(J) \subseteq \mathcal{C}(p(J))$; otherwise, assuming $J \neq \emptyset$ and $J \neq \{0\}$, necessarily $b \in p(J)$, and:
- if $0 \notin J$ and $0 \notin \mathcal{C}(J)$, then $p(\mathcal{C}(J)) = p(J) = \{b\} \subseteq \mathcal{C}(p(J))$;
- if $0 \notin J$ and $0 \in \mathcal{C}(J)$, then $p(\mathcal{C}(J)) = \{a, b\} = \mathcal{C}(\{b\}) = \mathcal{C}(p(J))$;
- if $0 \in J$, then $p(\mathcal{C}(J)) \subseteq \{a, b\} = \mathcal{C}(p(J))$.

We saw that Euclidean paths may not yield the expected results in quasi-discrete closure spaces. On the other hand, graph-theoretical and quasi-discrete paths coincide.

Lemma 2.33. *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.*

Note that, in particular, in Example 2.32 there is no *quasi-discrete* path rooted in a and passing by b, whereas there are quasi-discrete paths rooted in b and passing by a (for

example, the path defined by $p(0) = b$ and $p(i > 0) = a$). Let us introduce the notion of connectedness that we use in this work.

Definition 2.34. Given a closure space $(X, \mathcal{C})$, set $A \subseteq X$ is *path-connected* if and only if for each $x, y \in A$ there is a path p and an index i such that $p(0) = x$, $p(i) = y$ and, for all $j \leq i$, $p(j) \in A$.

Note that, for quasi-discrete closure spaces, by Lemma 2.33, Definition 2.34 coincides with the usual notion of *strong connectedness* in graph theory.

Remark 2.35. It is worth mentioning that connectedness can be also borrowed from topology, resorting to the notion of *separation*. Formally, let $(X, \mathcal{C})$ be a closure space. Two sets $A_1, A_2 \subseteq X$ are *separated* if and only if $\mathcal{C}(A_1) \cap A_2 = \emptyset = A_1 \cap \mathcal{C}(A_2)$. Note that separated sets are also disjoint, since for all sets A, we have $A \subseteq \mathcal{C}(A)$. Thus, there is no explicit requirement that A_1 and A_2 are disjoint. Set $A \subseteq X$ is *connected* if and only if there are no non-empty, separated sets $A_1, A_2 \subseteq X$ such that $A = A_1 \cup A_2$. In the case of topological spaces, the difference between this definition and path connectedness is widely known. There is a difference also in quasi-discrete closure spaces. A quasi-discrete closure space which is connected, but not path-connected is the space $(\{1, 2, 3\}, \mathcal{C}_R)$, where $R = \{(1, 2), (3, 2)\}$. By Lemma 2.33 there is no path from 1 to 3; however, it is not possible to find two non-empty, separated sets A_1, A_2 with $X = A_1 \cup A_2$. The only possible choices, recalling that separated sets must be disjoint, are $A_1 = \{1, 2\}, A_2 = \{3\}$, with $\mathcal{C}(A_2) \cap A_1 = \{2\}$, $A_1 = \{1\}, A_2 = \{2, 3\}$, with $\mathcal{C}(A_1) \cap A_2 = \{2\}$, and $A_1 = \{1, 3\}, A_2 = \{2\}$ with $\mathcal{C}(A_1) \cap A_2 = \{2\}$.

3. Spatial logics for closure spaces

In this section we present SLCS: a *Spatial Logic for Closure Spaces*, that we first proposed in [CLLM14a]. The logic 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 [CLLM14a], 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. In the present paper we extend SLCS with an additional binary operator, $\mathcal{P}$, used to model *propagation*, and propose a new interpretation for $\mathcal{U}$, based on the notion of paths that we introduced in Section 2.4. In order to avoid confusion, we call the newly defined connective *surrounded*, and use the symbol $\mathcal{S}$. Operator $\mathcal{S}$ coincides with $\mathcal{U}$ in the case of quasi-discrete closure spaces, and enhances it by also providing an intuitively meaningful interpretation in the case of continuous (e.g. Euclidean) spaces. The proposed spatial logic combines these new operators with standard boolean operators. Assume a finite or countable set AP of *atomic propositions*.

Definition 3.1. The syntax of SLCS is defined by the grammar in Figure 5, where a ranges over AP.

In 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, and $\mathcal{P}$ is the *propagation* operator. From now on, with a small overload of notation, we let Φ denote the set of SLCS formulas. We shall now define the interpretation of formulas.

$$
\begin{array}{rcll}
\Phi & ::= & a & [\textsc{Atomic proposition}] \\
 & | & \top & [\textsc{True}] \\
 & | & \neg\Phi & [\textsc{Not}] \\
 & | & \Phi \wedge \Phi & [\textsc{And}] \\
 & | & \mathcal{N}\Phi & [\textsc{Near}] \\
 & | & \Phi\,\mathcal{S}\,\Phi & [\textsc{Surrounded}] \\
 & | & \Phi\,\mathcal{P}\,\Phi & [\textsc{Propagation}]
\end{array}
$$

Figure 5: SLCS syntax

Definition 3.2. 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.

Definition 3.3. Satisfaction $\mathcal{M}, x \models \phi$ of formula $\phi \in \Phi$ at point $x \in X$ in model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$ is defined by induction on the structure of terms, by the equations in Figure 6.

$$
\begin{array}{lcll}
\mathcal{M}, x & \models & a \in AP & \iff x \in \mathcal{V}(a) \\
\mathcal{M}, x & \models & \top & \iff \mathit{true} \\
\mathcal{M}, x & \models & \neg\phi & \iff \mathcal{M}, x \not\models \phi \\
\mathcal{M}, x & \models & \phi_1 \wedge \phi_2 & \iff \mathcal{M}, x \models \phi_1 \text{ and } \mathcal{M}, x \models \phi_2 \\
\mathcal{M}, x & \models & \mathcal{N}\phi & \iff x \in \mathcal{C}(\{y \in X | \mathcal{M}, y \models \phi\}) \\
\mathcal{M}, x & \models & \phi_1\,\mathcal{S}\,\phi_2 & \iff \mathcal{M}, x \models \phi_1 \wedge \forall p : x \rightsquigarrow \infty . \forall l . \mathcal{M}, p(l) \models \neg\phi_1 \\
 & & & \qquad \implies \exists k . 0 < k \leq l . \mathcal{M}, p(k) \models \phi_2 \\
\mathcal{M}, x & \models & \phi_1\,\mathcal{P}\,\phi_2 & \iff \mathcal{M}, x \models \phi_2 \wedge \exists y . \exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty . \mathcal{M}, y \models \phi_1 \wedge \\
 & & & \qquad \forall i . 0 < i < l \implies \mathcal{M}, p(i) \models \phi_2
\end{array}
$$

Figure 6: SLCS semantics

Atomic propositions and boolean connectives have the expected meaning. For formulas of the form $\phi_1\,\mathcal{S}\,\phi_2$, the basic idea is that point x satisfies $\phi_1\,\mathcal{S}\,\phi_2$ whenever there is "no way out" from 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 $\mathit{green}\,\mathcal{S}\,\mathit{blue}$. A point x satisfies $\phi_1\,\mathcal{P}\,\phi_2$ 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.

In Figure 7, we present some derived operators. Besides standard logical connectives, the logic can express the *interior* ($\mathcal{I}\phi$), the *boundary* ($\delta\phi$), the *interior boundary* ($\delta^-\phi$) and the *closure boundary* ($\delta^+\phi$) of the set of points satisfying formula ϕ. Moreover, by appropriately using the *surrounded* operator, operators concerning *reachability* ($\phi_1\,\mathcal{R}\,\phi_2$), *global satisfaction* ($\mathcal{E}\,\phi$, *everywhere* ϕ) and *possible satisfaction* ($\mathcal{F}\phi$, *somewhere* ϕ) can be derived. Finally we define the $\mathcal{A}$ connective, expressing that ϕ_2 keeps x "apart" from ϕ_1. More explanation is provided below.

$$
\begin{array}{lcl@{\qquad}lcl}
\bot & \triangleq & \neg\top & \phi_1 \vee \phi_2 & \triangleq & \neg(\neg\phi_1 \wedge \neg\phi_2) \\
\mathcal{I}\phi & \triangleq & \neg(\mathcal{N}\neg\phi) & \delta\phi & \triangleq & (\mathcal{N}\phi) \wedge (\neg\mathcal{I}\phi) \\
\delta^{-}\phi & \triangleq & \phi \wedge (\neg\mathcal{I}\phi) & \delta^{+}\phi & \triangleq & (\mathcal{N}\phi) \wedge (\neg\phi) \\
\phi_1 \,\mathcal{R}\, \phi_2 & \triangleq & \neg((\neg\phi_2)\,\mathcal{S}(\neg\phi_1)) & \mathcal{E}\,\phi & \triangleq & \phi\,\mathcal{S}\,\bot \\
\mathcal{F}\phi & \triangleq & \neg\,\mathcal{E}(\neg\phi) & \phi_1 \,\mathcal{A}\, \phi_2 & \triangleq & \neg(\phi_1\,\mathcal{P}(\neg\phi_2))
\end{array}
$$

Figure 7: Some SLCS derived operators

Proposition 3.4. *We have that:*

(1) $\mathcal{M}, x \models \phi_1 \,\mathcal{R}\, \phi_2$ *if and only if there is* $p : x \rightsquigarrow \infty$ *and* k *such that* $\mathcal{M}, p(k) \models \phi_2$ *and for each* j *with* $0 < j \leq k$, *we have* $\mathcal{M}, p(j) \models \phi_1$*;*

(2) $\mathcal{M}, x \models \phi_1 \,\mathcal{A}\, \phi_2$*, if and only if* $\mathcal{M}, x \models \phi_2$ *or for any* y *such that* $\mathcal{M}, y \models \phi_1$*, and for any* $p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty$*, there exists* i *such that* $0 < i < l$ *and* $\mathcal{M}, p(i) \models \phi_2$*.*

(3) $\mathcal{M}, x \models \mathcal{E}\,\phi_1$ *if and only if for each* $p : x \rightsquigarrow \infty$ *and* $i \in \mathbb{N}$, $\mathcal{M}, p(i) \models \phi_1$*;*

(4) $\mathcal{M}, x \models \mathcal{F}\phi_1$ *if and only if there is* $p : x \rightsquigarrow \infty$ *and* $i \in \mathbb{N}$ *such that* $\mathcal{M}, p(i) \models \phi_1$*.*

Note that point x satisfies $\phi_1 \,\mathcal{R}\, \phi_2$ if and only if either ϕ_2 is satisfied by x or there exists a sequence of points after x, all satisfying ϕ_1, leading to a point satisfying both ϕ_2 and ϕ_1. In the second case, it is not required that x itself satisfies ϕ_1. For instance, both *red* and *green* nodes in Figure 2 satisfy $(\mathit{white} \vee \mathit{blue}) \,\mathcal{R}\, \mathit{blue}$, 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. We can define this notion as a derived operator as follows:

$$\phi_1 \,\mathcal{T}\, \phi_2 \triangleq \phi_1 \wedge ((\phi_1 \vee \phi_2) \,\mathcal{R}\, \phi_2)$$

Note also that ϕ_2 is occurring also in the first argument of $\mathcal{R}$. This is because satisfaction of $\phi_1 \,\mathcal{R}\, \phi_2$ requires that the final node on the path satisfies both ϕ_1 and ϕ_2.

A point x satisfies $\mathcal{M}, x \models \phi_1 \,\mathcal{A}\, \phi_2$ if it satisfies ϕ_2 or every path from a point y satisfying ϕ_1 to x passes by a point satisfying ϕ_2, located between y and x. For instance, with reference to Figure 2, let us consider $\mathit{yellow} \,\mathcal{A}\, \mathit{red}$, that is $\neg(\mathit{yellow} \,\mathcal{P}\, \neg \mathit{red})$. Note that $\mathit{yellow} \,\mathcal{P}\, \neg \mathit{red}$ is satisfied by the yellow points in the figure: for each yellow point x, let y be any yellow point (even x itself) and p a path starting from y and passing by x staying in the yellow area. Furthermore, points that are not yellow do not satisfy $\mathit{yellow} \,\mathcal{P}\, \neg \mathit{red}$ by definition of $\mathcal{P}$. Therefore, $\mathit{yellow} \,\mathcal{A}\, \mathit{red}$ is satisfied by all other points in the figure, including the red ones. Furthermore, all white nodes in the figure satisfy both $\mathit{yellow} \,\mathcal{A}\, \mathit{red}$ and $\mathit{green} \,\mathcal{A}\, \mathit{blue}$.

It is worth noting that in some situations, operators dealing with paths in opposite directions may be inter-expressible. However, an appropriate formalisation of such kinds of axioms, and the study of the associated classes of closure models, is left for future work.

We conclude this section by restricting our attention to *quasi discrete closure models*, i.e. *closure models* that are originated from *quasi discrete closure spaces*, in order to compare Definition 3.3 with the interpretation of $\mathcal{S}$ studied in [CLLM14a].

Definition 3.5. A *quasi discrete closure model* is a pair $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$ consisting of a quasi discrete 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.

Example 3.6. For $k, h \in \mathbb{N}$, let $\mathbb{N}^2_{k,h}$ be the set $\{(i, j) \in \mathbb{N} \times \mathbb{N} \mid i \in [1, k] \wedge j \in [1, h]\}$. A *digital image* of size $k \times h$, on finite set of *colours* C, is a function $f : \mathbb{N}^2_{k,h} \to C$, assigning a colour to each point of a finite rectangle in $\mathbb{N}^2$. Such an image gives rise to the quasi-discrete closure space $(\mathbb{N}_{k,h}, \mathcal{C}_{4adj})$, where

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

Furthermore, we also define the closure model $((\mathbb{N}_{k,h}, \mathcal{C}_{4adj}), \mathcal{V})$ with atomic propositions in C, where $\mathcal{V}(c \in C) = \{(i, j) \in \mathbb{N}^2_{k,h} \mid f(i, j) = c\}$.

In words, such closure model is based on a regular grid, where each pixel, except those on the borders, has four neighbours, corresponding to the directions right, left, up and down. On top of this space, atomic propositions are interpreted as the colours of pixels.

In [CLLM14a], we introduced the *spatial until* operator $\phi_1 \mathcal{U} \phi_2$, with a similar intended meaning as $\mathcal{S}$. The main difference is that the definition of $\mathcal{U}$ requires existence of a set of points satisfying ϕ_1, having closure boundary satisfying ϕ_2. The definitions of $\mathcal{S}$ and $\mathcal{U}$ coincide in the case of quasi-discrete spaces (see Theorem 3.7). As we will see, the definition using paths behaves in a more natural way for topological spaces. First, we compare the interpretation of $\mathcal{S}$ given in [CLLM14a] with Definition 3.3.

Theorem 3.7. *In a* quasi-discrete *closure model $\mathcal{M}$: $\mathcal{M}, x \models \phi_1 \mathcal{S} \phi_2$ according to Definition 3.3 if and only if $\mathcal{M}, x \models \phi_1 \mathcal{U} \phi_2$ according to* [CLLM14a]*, namely, there is $A \subseteq X$ such that $x \in A$, and $\forall y \in A. \mathcal{M}, y \models \phi_1$, and $\forall z \in \mathcal{B}^+(A). \mathcal{M}, z \models \phi_2$.*

We conclude this section by showing two examples where the definition of [CLLM14a] behaves in a counter-intuitive way, whereas the definition using paths works as expected.

Example 3.8. We define two models based on the Euclidean topology over $\mathbb{R}^2$, seen as a closure space $(\mathbb{R}^2, \mathcal{C})$. We use propositions b, w, g, depicted in Figure 8 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: $\mathcal{V}_1(b) = H \cup H^<$, $\mathcal{V}_1(w) = \mathbb{R}^2 \setminus \mathcal{V}_1(b)$, $\mathcal{V}_1(g) = \emptyset$, $\mathcal{V}_2(b) = \mathcal{V}_1(b)$, $\mathcal{V}_2(w) = H^\geq$, $\mathcal{V}_2(g) = \mathbb{R}_2 \setminus (H \cup H^< \cup H^\geq)$. Let $x \in H$. Clearly, 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. The expectation is that $b \mathcal{U} w$ 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 $\mathcal{M}_2, x \models b \mathcal{U} w$ 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.

4. The collective spatial logic CSLCS

So far, the properties expressed by our logic refer to points in space, when considered *individually*. However, when looking at space, it is also natural to formulate properties of *sets* of points, considered as a *collective* entity. As we shall see, our notion of collectivity is that of a set of points that are inter-reachable by paths in the whole space. Therefore, not only connected sets are of interest to our logic, but also sets of isolated points or

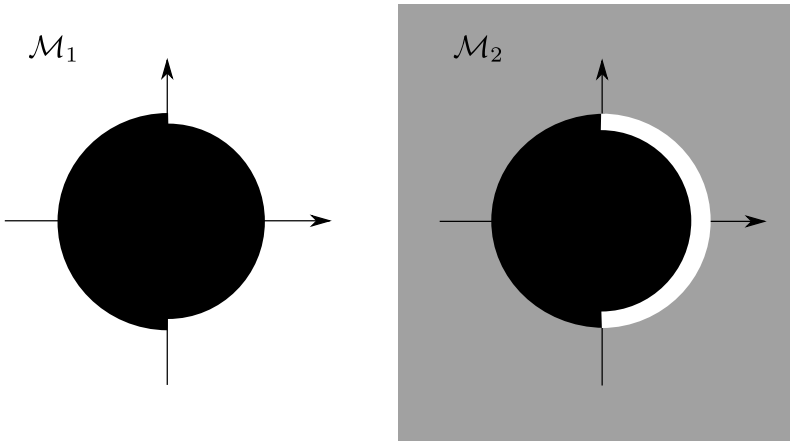

Figure 8: Two continuous closure models (boundaries are deliberately represented as very thick, but the reader should think of them as infinitely thin).

components, that are subsets of path-connected sets satisfying given properties. Other logics predicating on sets of points include the family of *region calculi* (see [KKWZ07] for a comprehensive overview), describing properties of *regular* sets, and using mereotopological boolean connectives (e.g., "part of", "boundary", and so on). Such logics characterise *regions* of space. We explicitly divert from this research line, because we aim at characterizing *local* properties of *points*, fitting in the tradition of modal logics, and relating individuals to the collectivity they live in. Our choice of collective operators is driven by this principle, and is modulated by the requirement of a computationally feasible model checking procedure.

Getting into detail, given a closure model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$, one may introduce "collective" formulas ψ (whose syntax and semantics will be clarified in the sequel) equipped with a *collective interpretation*, assigning a boolean valuation to the problem $\mathcal{M}, A \models \psi$ for each *set* of points $A \subseteq X$. We define the *collective spatial logic of closure spaces* CSLCS, which is interpreted on closure models. The logic has a *collective* fragment and an *individual* fragment. The collective fragment is evaluated on *subsets* of the set of points of the space. The individual fragment, which is evaluated on single points, is the logic SLCS defined in Section 3.

Definition 4.1. Fix a set AP of *atomic propositions*. The syntax of formulas is defined by the grammar in Figure 9, where a ranges over AP. •

We deliberately use the same syntax for boolean connectives both in the individual and the collective fragment, as usage of either fragment is always clear from the context. Boolean operators are standard. The novel operators we propose are the *share* connective and the *group* connective. Let ϕ be an individual formula, and ψ a collective formula. Informally, $\phi \prec\!\!\prec \psi$ (read: ϕ *share* ψ) is satisfied by set A when the subset of points of A satisfying the individual property ϕ also satisfies the collective property ψ. Formula $\mathcal{G}\phi$ holds on set A when its elements belong to a *group*, that is, a possibly larger, path-connected set of points, all satisfying the individual formula ϕ.

The satisfaction relation of the logic for each collective formula ψ is given in the form $\mathcal{M}, A \models_C \psi$, where $\mathcal{M}$ is a closure model (see Definition 3.2), and $A \subseteq X$ is a set of points.

Collective formulas

$$
\begin{array}{rcll}
\Psi & ::= & \top & [\textsc{True}] \\
 & | & \neg\Psi & [\textsc{Not}] \\
 & | & \Psi \wedge \Psi & [\textsc{And}] \\
 & | & \Phi \prec \Psi & [\textsc{Share}] \\
 & | & \mathcal{G}\Phi & [\textsc{Group}]
\end{array}
$$

Individual formulas

$$
\begin{array}{rcll}
\Phi & ::= & a & [\textsc{Atomic proposition}] \\
 & | & \top & [\textsc{True}] \\
 & | & \neg\Phi & [\textsc{Not}] \\
 & | & \Phi \wedge \Phi & [\textsc{And}] \\
 & | & \mathcal{N}\Phi & [\textsc{Near}] \\
 & | & \Phi\,\mathcal{P}\,\Phi & [\textsc{Propagation}] \\
 & | & \Phi\,\mathcal{S}\,\Phi & [\textsc{Surrounded}]
\end{array}
$$

Figure 9: CSLCS syntax.

Definition 4.2. Given a model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V}))$, and $A \subseteq X$, collective satisfaction $\models_C$ is given by the inductive definition below, where $\models$ is the individual satisfaction relation of Definition 3.3:

$$
\begin{array}{lll}
\mathcal{M}, A \models_C \top & & \\
\mathcal{M}, A \models_C \neg\psi & \iff & \mathcal{M}, A \not\models_C \psi \\
\mathcal{M}, A \models_C \psi_1 \wedge \psi_2 & \iff & \mathcal{M}, A \models_C \psi_1 \text{ and } \mathcal{M}, A \models_C \psi_2 \\
\mathcal{M}, A \models_C \phi \prec \psi & \iff & \mathcal{M}, \{x \in A \mid \mathcal{M}, x \models \phi\} \models_C \psi \\
\mathcal{M}, A \models_C \mathcal{G}\phi & \iff & \exists B \subseteq X. A \subseteq B \wedge B \text{ is path-connected} \wedge \\
 & & \forall z \in B. \mathcal{M}, z \models \phi
\end{array}
$$

The definition of $\mathcal{G}$ requires the existence of a set B which is possibly larger than A. The intuition is that the elements of A are part of a larger "collective", consisting of elements satisfying ϕ. We consider variants of connectedness as the most basic forms of *collective* and *spatial* property. In particular, we use path-connectedness, in line with the path-based interpretation of SLCS. Connectedness is "collective" in the sense that it is not merely determined by a property of the singletons composing a set, and it is not even preserved in subsets of a connected set. On the other hand, even though one could imagine all sorts of collective predicates on a model, we focus on (path-)connectedness, as it is completely determined by the structure of a closure space. For this reason, we consider it a fundamental collective property, deserving special treatment in the field of spatial logics, akin to the notion of transition in models of modal logics. Due to the restrictions that we introduce (mainly the strict layering of the collective and individual fragments) the logic CSLCS can be automatically verified at a computational cost which is comparable to that of SLCS. Using CSLCS one is able to check that given individuals are in the *same* area of space, and they share specific properties. Informally (and depending on the chosen closure model), this idea can be interpreted, for example, as: the fact that certain individuals are able to connect and act as a group; that they may follow the same route to reach a goal; that they are located all together in a protected environment; etc. Below, we develop this concept by the means of some derived operators. In Section 7 we provide some examples.

Definition 4.3. The following derived operators may be defined, where ψ_1 and ψ_2 are collective formulas, and ϕ is an SLCS formula:

$$\begin{array}{lcll} \bot & \triangleq & \neg\top & [\textsc{False}] \\ \psi_1 \vee \psi_2 & \triangleq & \neg((\neg\psi_1) \wedge (\neg\psi_2)) & [\textsc{Or}] \\ \forall\phi & \triangleq & \neg\phi \prec \mathcal{G}\bot & [\textsc{Forall, Individually}] \\ \exists\phi & \triangleq & \neg(\forall\neg\phi) & [\textsc{Exists}] \\ \emptyset & \triangleq & \forall\bot & [\textsc{Empty}] \end{array}$$

The definition of $\forall$ uses the fact that the only set A such that $\mathcal{M}, A \models \mathcal{G}\bot$ is the empty set, which is trivially path-connected. This is made formal by the following lemma.

Lemma 4.4. *We have:*
(1) $\mathcal{M}, A \models_C \forall\phi$ *if and only if* $\forall x \in A. \mathcal{M}, x \models \phi$*;*
(2) $\mathcal{M}, A \models_C \exists\phi$ *if and only if* $\exists x \in A. \mathcal{M}, x \models \phi$*;*
(3) $\mathcal{M}, A \models_C \emptyset$ *if and only if* $A = \emptyset$*.*

The $\forall$ and $\exists$ connectives also exist in the classical topological logic $\mathcal{S}4_u$ (see [KKWZ07]); additionally, CSLCS provides the possibility to classify subsets, instead of whole models. However, *global satisfaction*, defined on models, is obtained as a side effect.

Definition 4.5. *Global* satisfaction is defined for each model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$ and collective formula ψ as $\mathcal{M} \models_G \psi \iff \mathcal{M}, X \models_C \psi$.

From now on, we will sometimes omit the subscripts C and G from the satisfaction relation, when clear from the context. Apart from the usual derived connectives, such as disjunction or logical implication, CSLCS can express some useful derived operators.

Definition 4.6. Define the following collective derived operators:

$$\begin{array}{lcll} \phi_1 \,\mathcal{CS}\, \phi_2 & \triangleq & \mathcal{G}(\neg\phi_2 \wedge (\phi_1 \,\mathcal{S}\, \phi_2)) & [\textsc{Collectively surrounded}] \\ \phi_1 \,\mathcal{CP}\, \phi_2 & \triangleq & \forall((\phi_1 \vee \phi_2) \wedge \neg(\phi_1 \wedge \phi_2)) \wedge & [\textsc{Collectively partitioned}] \\ & & (\phi_1 \prec (\phi_1 \,\mathcal{CS}\, \phi_2)) \wedge (\phi_2 \prec (\phi_2 \,\mathcal{CS}\, \phi_1)) & \end{array}$$

A set A satisfies $\mathcal{M}, A \models \phi_1 \,\mathcal{CS}\, \phi_2$ if and only if the points in A satisfy ϕ_1, and are "collectively" surrounded by a set of points satisfying ϕ_2. More precisely, using the connective $\mathcal{G}$, it is required that a path-connected set B including A exists, with all points of B satisfying $\phi_1 \,\mathcal{S}\, \phi_2$, but not ϕ_2. Not only there can be no path rooted in B and leaving ϕ_1 without passing by ϕ_2, but also, noting that all the elements of B satisfy $\phi_1 \wedge \neg\phi_2$, such set B must be a path-connected component of $\neg\phi_2$, the elements of which are surrounded in the sense of SLCS by points satisfying ϕ_2.

For the $\mathcal{CP}$ connective, we look at its global interpretation. The statement $\mathcal{M} \models \phi_1 \,\mathcal{CP}\, \phi_2$ expresses that all the points of the space satisfy either ϕ_1 or ϕ_2, that *all* the points satisfying ϕ_1 can be connected to each other, forming a set of points satisfying ϕ_1 and surrounded by points satisfying ϕ_2, and vice-versa. The sets of points satisfying ϕ_1 is path-connected, and so is the set satisfying ϕ_2. For example, the model in the left-hand side of Figure 10 satisfies $\mathit{red} \,\mathcal{CP}\, \mathit{blue}$ while the model in the right-hand-side of the figure does not satisfy the same formula.

Figure 10: The model on the left satisfies $red\,\mathcal{CP}\,blue$; the one on the right does not.

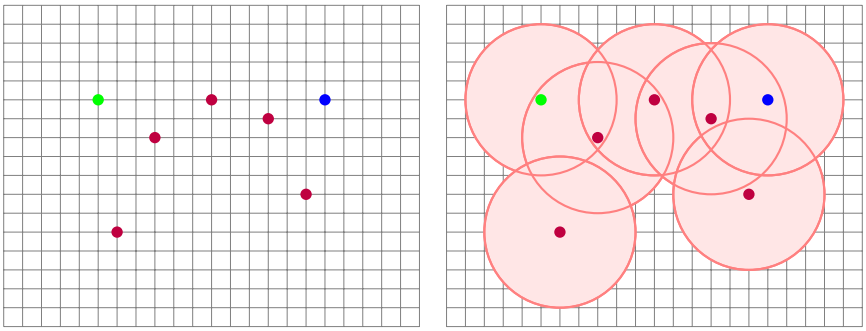

Figure 11: A graphical representation of Example 5.1.

5. Example: emergency evacuation

In this section we show some examples of interpreting SLCS and CSLCS on quasi-discrete closure spaces. First, starting from our running example, let us define a closure space to provide a simple model of short-range communication.

Example 5.1. Let us consider again the closure space presented in Example 2.6. This closure space can be used to model a network of agents distributed over a two-dimensional physical space, that communicate via wireless devices having fixed communication radius δ. In the left hand side of Figure 11 a graphical representation of such model is provided. There *green*, *purple* and *blue* dots identify different kinds of agents located in the space. Let us consider the colours as atomic propositions.

The set $\mathcal{C}_\delta(green \cup purple \cup blue)$ consists of points in $\mathbb{R}^2$ that are in the communication range of at least one agent, represented by the pink area in the right-hand side of Figure 11. Suppose that the *green* agent of our example is the source of some relevant information, which is meant to be transmitted from the *green* device to the other devices that are reachable after some *hops*. The set of devices that can receive the information sent by the *green* device is characterised, using the propagation operator, by the formula $green\,\mathcal{P}(purple \cup blue)$, satisfied by the black points in Figure 12.

Taking advantage of both Example 3.6 (interpreting digital images as closure models) and Example 5.1, we will now set up a more complex closure space, comprising a communication layer, with closure determined by communication ranges, and a physical layer, with closure determined by the structure of a regular grid. The two layers are linked by a binary relation. On top of this set-up, we will discuss the interpretation of some example properties, assuming that a set of agents (modelled by appropriate atomic propositions) is distributed in the physical layer.

Example 5.2. Recall from Example 3.6 that digital images can be treated as finite quasi-discrete closure models. Consider one such model, with underlying space $(X, \mathcal{C}_{4adj})$, with

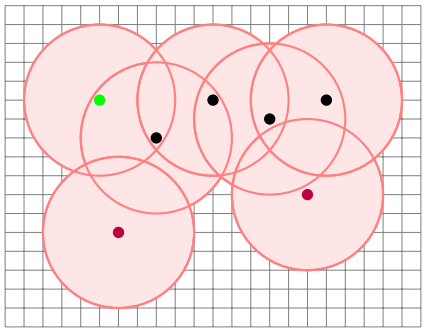

Figure 12: In *black* the devices that can receive data from the *green* device.

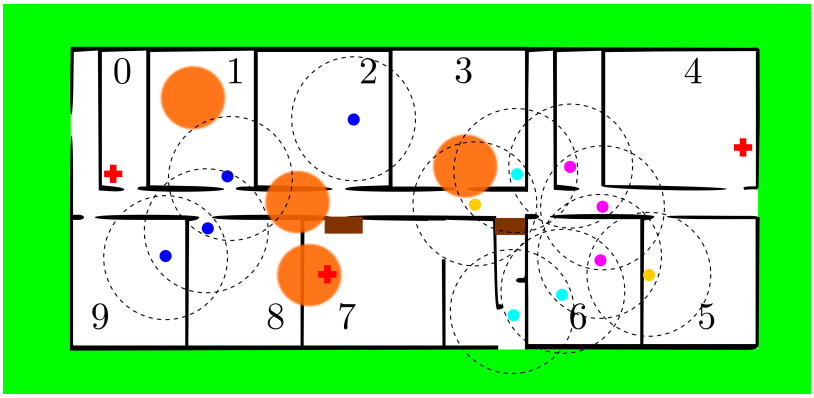

Figure 13: A representation of agents in a building in an emergency condition.

$X \subset \mathbb{N}^2$. In this example, we will use a digital image representing a portion of a two-dimensional physical space; therefore, each point of the image is also mapped to a *position*, or *coordinate*, in the Euclidean space $\mathbb{R}^2$, giving rise to a function $map : X \to \mathbb{R}^2$. Let Y be the finite image of the function map. Assume X and Y are disjoint, for simplicity. Let pos be the graph of the function map, that is, the set of pairs $\{(x, y) \in X \times Y \mid map(x) = y\}$. In a similar way as in Example 5.1, fix a communication range δ, and introduce the relation $R_\delta \subseteq \mathbb{R}^2$ from Example 2.24. Then, let $R'_\delta = R_\delta \cap Y^2$ be the restriction of R_δ to the image of the function map. Consider the set $Z = X \cup Y$. Define the quasi-discrete closure space $(Z, \mathcal{C}_R)$ using the relation

$$R \triangleq 4adj \cup pos \cup R'_\delta$$

The closure space $(Z, \mathcal{C}_R)$ can be thought of as "two-layered". One layer is the digital image, the other one is a finite subset of $\mathbb{R}^2$ equipped with the closure $\mathcal{C}_\delta$ restricted to Y, in a similar way to Example 2.6 . The two layers are linked by the relation pos; note that each position in Y is thus "close", in the sense of the operator $\mathcal{N}$, to a point of the digital image. By this, as we shall see, logic formulas can simultaneously predicate on proximity in the image, acting as a "physical" layer, where proximity means adjacency in space, and in Euclidean coordinates, acting as a "communication" layer, where proximity is based on distance. We will also consider a set of agents, first-aid facilities, obstructions, and dangerous areas, formalised as atomic propositions, giving rise to a quasi-discrete closure *model*.

Before making this idea formal, we look at a picture of an instance of such construction, in Figure 13. The digital image in the background, the points of which form the set X, represents the map of a building at a specific instant in time, where an emergency situation occurs (note that rooms have been numbered for reader's convenience, but we are not considering numbers, graphically, as part of the underlying map). The *white* points form the areas where agents can walk. Some of the white points, however, are covered by obstructions, painted in brown, or are in the range of some source of hazard. Hazardous areas are painted in semi-transparent orange. The green points are a safe area, accessible via exit doors. Some white points are also part of areas where first aid is available, which are represented by a red cross. The walls are painted in black. Coloured (blue, cyan, purple, yellow) dots represent agents, with their communication range (dashed circles). The set Y is the set of actual coordinates of the points in space denoted by pixels of the digital image.

We define a valuation function $\mathcal{V}$, obtaining the quasi-discrete model $((Z, \mathcal{C}_R), \mathcal{V})$. Atomic propositions are the colours *white*, *black*, *green*, *red*, *blue*, *cyan*, *purple*, *yellow*, *brown*, *agent*, *danger*, and *coord*. Function $\mathcal{V}$ is such that each point in the image satisfies its own colour. Proposition *danger* is true only at points in the image under the orange semi-transparent circles. Each point may satisfy more than one atomic proposition; in particular, points under the orange circles also satisfy other atomic propositions. Agents are represented by additionally colouring points of X in blue, cyan, purple, or yellow. Points that satisfy *red* or *brown* also satisfy *white*, as in principle these are areas where it is possible to walk, even though there is an obstruction in the current situation. Points of Y satisfy just one predicate, namely *coord*, and are not represented in Figure 13. In addition, no other point in Z satisfies predicate *coord*. Finally, define the short-hands $\mathit{obstacle} \triangleq \mathit{black} \vee \mathit{brown} \vee \mathit{danger}$, $\mathit{agent} \triangleq \mathit{blue} \vee \mathit{cyan} \vee \mathit{purple} \vee \mathit{yellow}$, and $\mathit{safe} \triangleq \mathit{white} \wedge \neg \mathit{obstacle}$.

In this situation, we suppose that groups of agents of the same colour are expected to address an emergency situation together. Agents must be able to reach both first-aid points and exit doors without passing by dangerous areas. Agents belonging to the same group should reach a first-aid point and the exit together with other members of the group; in case an agent is isolated from her group, an agent of another group must be able to reach first aid, and then rescue her.

We remark that, for simplicity, when dealing with paths concerning agents, we do not consider the cases in which an agent may exit and re-enter the building through a different access, passing by the green area[4]. In the remainder of this section, we present some example properties, and their interpretation in the situation of Figure 13. First, recall the definition of the derived operator $\phi_1 \mathcal{T} \phi_2 \triangleq \phi_1 \wedge ((\phi_1 \vee \phi_2) \mathcal{R} \phi_2)$. Point x satisfies $\phi_1 \mathcal{T} \phi_2$ whenever it satisfies ϕ_1 and there is a path p, and an index i, with $p(0) = x$, such that, for all $j \in (0, i)$, point $p(j)$ satisfies $\phi_1 \vee \phi_2$, and point $p(i)$ satisfies ϕ_2. Informally speaking, we may say that $\mathcal{T}$ expresses reachability in space from a point satisfying formula ϕ_1 to a point satisfying ϕ_2, only passing by points satisfying ϕ_1 or ϕ_2.

Example 5.3. There may be safe points, with no escape route. This is defined as the formula

$$\phi_1 \triangleq \mathit{safe}\, \mathcal{S}\, \mathit{obstacle}$$

satisfied by the white points in Room 3.

[4]Depending on the application domain, one may take into account agents that exit and re-enter the building by using another set of logical properties (the logic easily distinguishes between these two different kinds of paths).

Example 5.4. The walking areas, from which a first-aid point can be safely reached, are classified by the derived operator $\mathcal{R}$. Consider the formula:

$$\phi_2 \triangleq \mathit{safe}\,\mathcal{T}\,(\mathit{red} \wedge \mathit{safe})$$

Points satisfying formula ϕ_2 are required to be *safe*, and furthermore, to be at the start of a path of *safe* points, leading to a point which is *red* and *safe*. In Figure 13, ϕ_2 is satisfied, among other points, by all the positions of agents, except those in rooms 3 and 7. That is, ϕ_2 is satisfied by those white points that are the start of a path that avoids obstacles (including dangerous areas), leading to safe first-aid facilities, while only traversing *white* points. Similarly, the points from which an exit may be reached are characterised by the formula

$$\phi_3 \triangleq \mathit{safe}\,\mathcal{T}\,\mathit{green}$$

which is satisfied by the blue, yellow, and violet points, but not by any cyan point (note that we are not considering the possibility of passing by the green area and re-enter the building, as we explained earlier). The points where first-aid facilities are located, and from where it is possible to safely reach an exit (all the red points in Figure 13), satisfy the formula

$$\phi_4 \triangleq (\mathit{red} \wedge \mathit{safe}) \wedge (\mathit{safe}\,\mathcal{T}\,\mathit{green})$$

Combining ϕ_2 and ϕ_4 one is then able to define the set of points from which one can safely walk to a first aid point and then to the exit. These points are identified by the formula

$$\phi_5 \triangleq \mathit{safe}\,\mathcal{T}\,\phi_4$$

For instance, the white points in Room 8, but not those in Room 7, satisfy ϕ_5.

We shall now introduce some collective formulas, that for simplicity are evaluated under the global interpretation of Definition 4.5.

Example 5.5. We can define a collective formula, parametrised by a colour, that is true whenever all agents of the given colour are connected in the communication layer of the model.

$$\phi_6(\mathit{colour}) = (\mathit{coord} \wedge \mathcal{N}\mathit{colour}) \prec \mathcal{G}(\mathit{coord} \wedge \mathcal{N}\mathit{agent})$$

In the definition of ϕ_6, note that $\mathcal{N}\mathit{colour}$ denotes the set of points that are *near* to a point satisfying *colour*. Such set is the union of the points in the digital image where the agents of the given colour are located, their neighbours in the digital image, and their coordinates in the communication layer. Therefore, when *colour* is the colour of an agent, the sub-formula $\mathit{coord} \wedge \mathcal{N}\mathit{colour}$ precisely identifies the coordinates in Y that are positions of agents in the group identified by *colour*. Such coordinates are required to be part of a larger set of points, which are *connected* in the communication layer, and also are positions of arbitrary agents, so that the communication flow required by the formula may also include agents of *different* colours. In the model of Figure 13, $\phi_6(\mathit{colour})$ holds for all the colours of agents, except blue.

Example 5.6. Agents of the same colour should be able to reach a first aid point, and then an exit, all together. We leave the colour as a parameter of the formula.

$$\phi_7(\mathit{colour}) = \mathit{colour} \prec \mathcal{G}\phi_5$$

In Figure 13, $\phi_7(\mathit{colour})$ holds for colours yellow and purple, but not cyan and blue.

Example 5.7. We shall now deal with rescuing of agents. An agent of a given colour can be rescued if there is an agent of a different colour that can reach her, after passing by a first-aid point, and the two can safely reach an exit. First consider formula $\phi_8(\mathit{colour})$, describing first-aid points that can be reached by an agent of a different colour than the given one (this is achieved by the sub-formula $\mathit{agent} \wedge (\neg \mathit{colour})$ below), by a safe route:

$$\phi_8(\mathit{colour}) \triangleq (\mathit{red} \wedge \neg \mathit{obstacle}) \wedge (\mathcal{N}((\mathit{agent} \wedge (\neg \mathit{colour}))\, \mathcal{P}\, \mathit{safe}))$$

Points satisfying $\phi_8(\mathit{colour})$ are *red* and not an obstacle, that is, they are safe first-aid locations. Furthermore, the definition of $\phi_8(\mathit{colour})$ also uses the $\mathcal{P}$ operator in order to guarantee that such points are directly connected (operator $\mathcal{N}$) to points that *can be reached*[5] from a point where an agent of a different colour is located, passing only through safe points. Thus, agents of a specific colour that can be rescued satisfy the formula

$$\phi_9(\mathit{colour}) \triangleq \mathit{agent} \wedge \phi_2 \wedge \mathcal{N}(\neg \mathit{obstacle} \wedge (\phi_8(\mathit{colour})\, \mathcal{P}\, \mathit{safe}))$$

We can also define a collective formula expressing that, for a given colour, either $\phi_7(\mathit{colour})$ holds, or all agents can be rescued:

$$\phi_{10}(\mathit{colour}) \triangleq \phi_7(\mathit{colour}) \vee \forall(\mathit{colour} \prec \phi_9(\mathit{colour}))$$

In our example model, $\phi_{10}(\mathit{blue})$ is true, whereas $\phi_{10}(\mathit{cyan})$ is false.

6. Spatial model checking

In this section we describe a model checking algorithm for SLCS and CSLCS. The algorithm is composed of two procedures, one for individual formulas, that is, the logic SLCS, and one for collective formulas, making use of the procedure for individual formulas. As we shall see, the procedure for individual formulas is a *global* model checking procedure for SLCS. Given model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$ and formula ϕ, the procedure returns the set $\{x \in X \mid \mathcal{M}, x \models \phi\}$. The procedure for collective formulas, on the other hand, is a *local* model checking algorithm, that is, given model $\mathcal{M}$, formula ψ and set of points A, it returns the boolean satisfaction value of $\mathcal{M}, A \models \psi$. We choose a local algorithm for the collective fragment, since enumeration of a set of subsets is a problem of inherent exponential complexity. Merely returning a result for a global model checking procedure would require some kind of symbolic description, which is left for future investigation.

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, $\neg\phi_1$ or $\phi_1 \wedge \phi_2$, the definition of $\texttt{Sat}(\mathcal{M}, \phi)$ is straightforward. To compute the set of points satisfying $\mathcal{N}\phi_1$, the closure operator $\mathcal{C}$ of the space is applied to the set of points satisfying ϕ_1. When ϕ is of the form $\phi_1\, \mathcal{S}\, \phi_2$, function `Sat` relies on the function `CheckSurr` defined in Algorithm 2. When ϕ is of the form $\phi_1\, \mathcal{P}\, \phi_2$, function `Sat` relies on the function `CheckProp` defined in Algorithm 3.

[5]Since the model of our example is symmetric, reachability in opposite directions may not make an actual difference. However, models similar to the one we are depicting may feature e.g., one way doors. We are not adding one-way links in our model, as we do not deem it necessary for illustrating the connectives of the logic, and it makes the formal definition of the underlying closure space less readable.

Function $\texttt{Sat}(\mathcal{M}, \phi)$
  Input: Finite, quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, formula ϕ
  Output: Set of points $\{x \in X \mid \mathcal{M}, x \models \phi\}$
  Match ϕ
    case $\top$: **return** X
    case p : **return** $\mathcal{V}(p)$
    case $\neg\phi_1$:
      let $P = \texttt{Sat}(\mathcal{M}, \phi_1)$
      return $X \setminus P$
    case $\phi_1 \wedge \phi_2$:
      let $P = \texttt{Sat}(\mathcal{M}, \phi_1)$
      let $Q = \texttt{Sat}(\mathcal{M}, \phi_2)$
      return $P \cap Q$
    case $\phi_1 \mathcal{P} \phi_2$:
      return `CheckProp` $(\mathcal{M},\phi_1,\phi_2)$
    case $\phi_1 \mathcal{S} \phi_2$:
      return `CheckSurr` $(\mathcal{M},\phi_1,\phi_2)$

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

Function `CheckSurr` $(\mathcal{M},\phi_1,\phi_2)$
  Input: Finite, quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, formulas ϕ_1, ϕ_2
  Output: Set of points $\{x \in X \mid \mathcal{M}, x \models \phi_1 \mathcal{S} \phi_2\}$
  var $V := \texttt{Sat}(\mathcal{M}, \phi_1)$
  let $Q = \texttt{Sat}(\mathcal{M}, \phi_2)$
  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.

Function `CheckProp` $(\mathcal{M},\phi_1,\phi_2)$
  Input: Finite, quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, formulas ϕ_1, ϕ_2
  Output: Set of points $\{x \in X \mid \mathcal{M}, x \models \phi_1 \mathcal{P} \phi_2\}$
  var $V := \texttt{Sat}(\mathcal{M}, \phi_1)$
  var $Q = \texttt{Sat}(\mathcal{M}, \phi_2)$
  var $T := \mathcal{C}_R(V) \cap Q$
  var $R := T$
  var $Q := Q \setminus T$
  while $T \neq \emptyset$ **do**
    var $T' := \emptyset$
    for $x \in T$ **do**
      $T' := T' \cup (Q \cap post(x))$
    $Q := Q \setminus T'$
    $R := R \cup T'$
    $T := T'$
  return R

Algorithm 3: Checking *propagation* formulas in a quasi-discrete closure space.

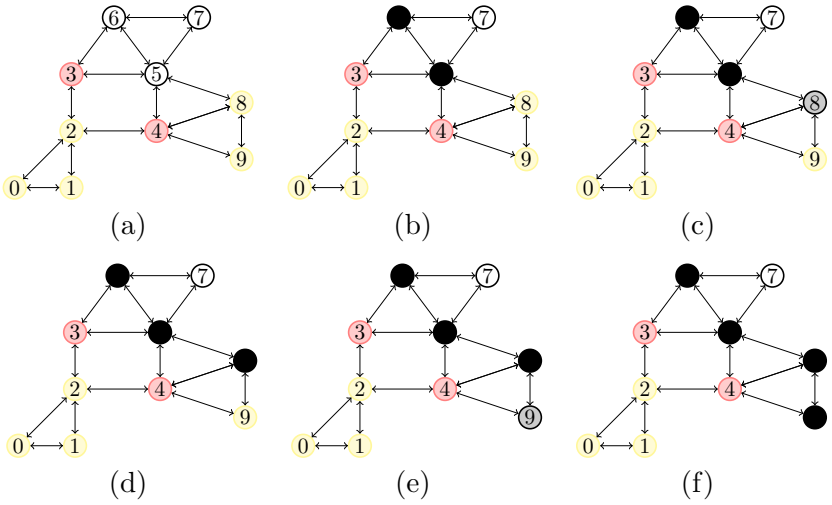

Figure 14: Model-checking *yellow* $\mathcal{S}$ *red*

Function `CheckSurr` takes as parameters a finite, quasi-discrete closure model $\mathcal{M}$, and two SLCS formulas ϕ_1 and ϕ_2. The function computes the set of points in $\mathcal{M}$ satisfying $\phi_1 \,\mathcal{S}\, \phi_2$. This is performed iteratively by removing from $V = \mathtt{Sat}(\mathcal{M}, \phi_1)$ points that we may intuitively call *bad*. More precisely, a point is *bad* if, in the underlying relation of the quasi-discrete closure model, there is a path rooted in it, reaching a point satisfying $\neg\phi_1$, without crossing any point satisfying ϕ_2. Let $Q = \mathtt{Sat}(\mathcal{M}, \phi_2)$ be the set of points in $\mathcal{M}$ satisfying ϕ_2. To identify the *bad* points in V the function `CheckSurr` performs a *backward search* from $T = \mathcal{B}^+(V \cup Q)$. Note that any path leaving $V \cup Q$ must pass through points in T. Moreover, T only contains points that satisfy neither ϕ_1 nor ϕ_2. Until T is empty, function `CheckSurr` first picks an element x in T and then removes from V the set of (bad) points N that can reach x in *one step*. To compute the set N we use the function $pre(x) = \{y \in X \mid (y, x) \in R\} = \{y \in X \mid x \in \mathcal{C}_R(\{y\})\}$. At the end of each iteration the variable T is updated by considering the set of newly discovered *bad points*. Note that such new bad points do not include "candidate bad points" that also satisfy ϕ_2. This is because any such point x satisfies both formulas, thus every path starting from x and reaching $\neg\phi_1$ also passes (trivially) by a point satisfying ϕ_2. The evolution of this algorithm is illustrated in an informal way in Figure 14 for the formula *yellow* $\mathcal{S}$ *red*. At the beginning we have that $V = \{0, 1, 2, 8, 9\}$ and $Q = \{3, 4\}$. Variable T is initialized to the external boundary of $V \cup Q$, that is the set $\{5, 6\}$. Points in T are the black ones in Figure 14 (b). In the next step, yellow points that are neighbours of black ones (coloured in grey in Figure 14 (c)) are removed from V and included in T (see Figure 14 (d)). This "refinement" step is iterated until a fixed point is reached. The remaining yellow points are those satisfying *yellow* $\mathcal{S}$ *red* (see Figure 14 (f)).

Function `CheckProp` takes as parameters a finite, quasi-discrete closure model $\mathcal{M}$, and two SLCS formulas ϕ_1 and ϕ_2. The function computes the set of points in $\mathcal{M}$ satisfying $\phi_1 \,\mathcal{P}\, \phi_2$. Such computation is performed iteratively via a breadth-first search that starts from all the points in $V = \mathtt{Sat}(\mathcal{M}, \phi_1)$ and that traverses only points that are in $Q = \mathtt{Sat}(\mathcal{M}, \phi_2)$. To select points at the next level, the function $post(x) = \{y | (x, y) \in R\} = \mathcal{C}_R(\{x\})$ is used. The evolution of this algorithm is illustrated in an informal way in Figure 15 for the formula

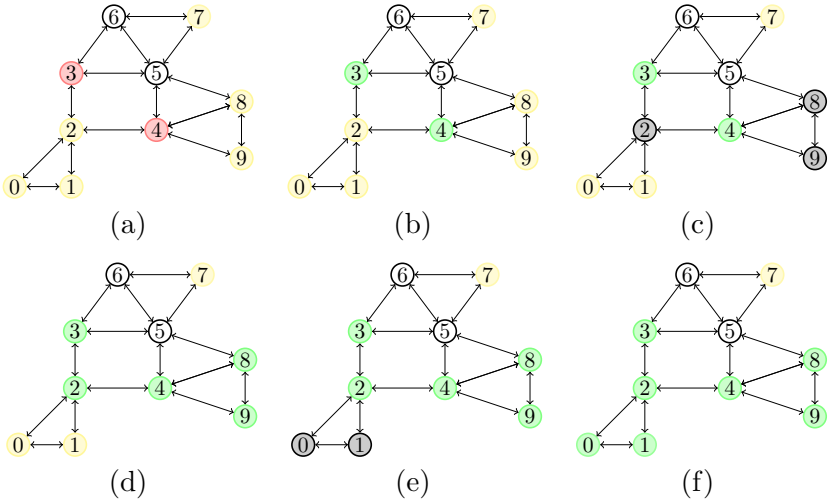

Figure 15: Model-checking *red* $\mathcal{P}$ *yellow*

red $\mathcal{P}$ *yellow*. First, all *red* points – which satisfy *red* $\mathcal{P}$ *yellow* – are included in the set R (the green points in Figure 15 (b)). In the next step, the algorithm selects all the yellow points that are neighbours of an element in R (Figure 15 (c) and Figure 15 (e)). These points are added to the set R until a fixed point is reached (see Figure 15 (f)). When the algorithm terminates, the points in R are exactly the ones satisfying the considered formula. These are the green points in Figure 15 (f).

The local model checking algorithm for CSLCS formulas is given in Algorithm 4. Function $\texttt{Sat}_C$ takes as input a finite, quasi-discrete model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$, a subset A of X and a collective formula ψ, and returns the truth value of $\mathcal{M}, A \models \psi$. The definition uses function `Sat` as defined above. The implementation of boolean operators is straightforward. The case for $\phi \prec\!\!\prec \psi$ uses the global model checker `Sat` for individual formulas to compute the set of points satisfying ϕ, and recursively checks if the intersection of such set with A satisfies ψ. The case for $\mathcal{G}\phi$ first performs some checks for corner cases of the definition, namely when A is the empty set (then $\mathcal{G}\phi$ is true), and when A is not included in the set of points satisfying ϕ (then $\mathcal{G}\phi$ is false). After this, a variant of the classical Tarjan's algorithm [Tar72] for computing strongly connected components is executed on the underlying graph of the space, starting from an arbitrary point of A. The pseudo-code for such procedure is reported in Algorithm 5.

More specifically, the difference between our algorithm and the classical procedure by Tarjan is that we only visit nodes in B (that is, the semantics of ϕ), and reachable from a chosen element x of A, whereas the classical procedure visits all the nodes of the graph. This choice is motivated by the fact that we do not need to collect all the strongly connected components, but only to determine whether there is a strongly connected component, in the subgraph determined by B, that contains A. In the algorithm, s is a stack; for simplicity we assume an operation $popUntil(s, x)$ that removes from a stack the most recently inserted elements including x, and returns the set of all such elements. Note that such set is only needed to compare it with A; this check can be efficiently implemented with a while loop that pops elements out of the stack and checks whether such elements belong to A, thus avoiding to store an additional set of possibly large size. Furthermore, ll is a *map* (the "low

link" array of Tarjan's algorithm), indexed by elements of X. We omit the details of its implementation; clearly, if X is enumerated by a contiguous subset of the natural numbers, a standard array can be used.

In order to address termination, complexity and correctness of our algorithms, we first define the notion of *size* of a formula.

Definition 6.1. For ϕ a SLCS formula, let $size(\phi)$ be inductively defined as follows:

- $size(\top) = size(p) = 1$
- $size(\neg\phi) = size(\mathcal{N}\phi) = 1 + size(\phi)$
- $size(\phi_1 \wedge \phi_2) = size(\phi_1 \,\mathcal{S}\, \phi_2) = 1 + size(\phi_1) + size(\phi_2)$

For ψ a CSLCS formula, let $size(\psi)$ be inductively defined as follows:

- $size(\top) = 1$
- $size(\neg\psi) = size(\mathcal{G}\psi) = 1 + size(\psi)$
- $size(\psi_1 \wedge \psi_2) = 1 + size(\phi_1) + size(\phi_2)$
- $size(\phi \prec \psi) = 1 + size(\phi) + size(\psi)$

Lemma 6.2. *For any finite quasi-discrete model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$ and SLCS formula ϕ of size k,* `Sat` *terminates in $\mathcal{O}(k \cdot (|X| + |R|))$ steps.*

Theorem 6.3. *For any finite quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$ and SLCS formula ϕ, $x \in \mathtt{Sat}(\mathcal{M}, \phi)$ if and only if $\mathcal{M}, x \models \phi$.*

Theorem 6.4. *For any finite, quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, formula ψ with $size(\psi) = k$, and $A \subseteq X$, we have $\mathtt{Sat}_C(\mathcal{M}, A, \psi) = True$ if and only if $\mathcal{M}, A \models \psi$, taking in the worst case $\mathcal{O}(k \cdot (|X| + |R|))$ steps.*

7. A model checker for SLCS and CSLCS

The algorithms described in Section 6 are available as a proof-of-concept tool[6]. The tool is implemented in OCaml[7], and can be invoked both as a global model checker for SLCS, or as a local model checker for CSLCS.

In the following we discuss a few examples showing how the tool can be used for identifying and analysing regions of interest of a digital image (e.g., a map, a medical image, a picture etc.), using spatial formulas. In this section, digital images are treated as finite, quasi-discrete models in the plane $\mathbb{N} \times \mathbb{N}$, equipped with the closure operator $4adj$ of Example 3.6. Other topologies can be readily implemented in the tool. In the case of images, the tool accepts as atomic propositions expressions that denote sets of colours, so that each point (x, y) satisfies precisely those expressions whose semantics includes the colour of the pixel at coordinates (x, y). The SLCS model checker, which implements a global algorithm, accepts a formula ϕ, a colour c and a digital image, and colours with c the points of the image satisfying ϕ. The CSLCS model checker, which is a local algorithm, implements both Definition 4.2 and Definition 4.5, accepting a collective formula ψ, and optionally[8]a set of points, and returning a boolean answer.

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

[7]See `http://ocaml.org`.

[8]If no points are specified, the whole space is considered for global satisfaction.

Function $\mathtt{Sat}_C(\mathcal{M}, A, \psi)$
  Input: Finite, quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, Set of points A, collective formula ψ
  Output: Truth value of $\mathcal{M}, A \models \psi$
  Match ψ
    case $\top$: **return** *True*
    case $\neg\psi$:
      let $R = \mathtt{Sat}_C(\mathcal{M}, A, \psi)$
      return not R
    case $\psi_1 \wedge \psi_2$:
      let $R = \mathtt{Sat}_C(\mathcal{M}, A, \psi_1)$
      let $S = \mathtt{Sat}_C(\mathcal{M}, A, \psi_2)$
      return R **and** S
    case $\phi \prec \psi_1$:
      let $B = \mathtt{Sat}(\phi) \cap A$
      return $\mathtt{Sat}_C(\mathcal{M}, B, \psi_1)$
    case $\mathcal{G}\phi$:
      if $(A = \emptyset)$ **return** *True*
      let $B = \mathtt{Sat}(\phi)$
      if $(A \not\subseteq B)$ **return** *False*
      let $x \in A$
      let $t = newCounter()$
      let $s = newStack()$
      let
        $ll = newMap(X, undefined)$

      return
        $\mathtt{Visit}(\mathcal{M},t,s,ll,A,B,x)$

Algorithm 4: Algorithm for the model checking problem of CSLCS

Function $\mathtt{Visit}$ $(\mathcal{M},t,s,ll,A,B,x)$
  Input: Finite, quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, counter t, stack s, vector ll, sets of points A, B, point x
  Output: Truth value or *undefined*, depending on the progress of the algorithm when this auxiliary function is called.
  var $isRoot := True$
  var $r := undefined$
  $push(s, x)$
  $ll[x] := increment(t)$
  for $y \in post(x) \cap B$ **do**
    if $(ll[y] = undefined)$
      $r :=$ $\mathtt{Visit}$ $(\mathcal{M},t,s,ll,A,B,y)$
    if $(r \neq undefined)$ **return** r
    if $(ll[x] > ll[y])$
      $ll[x] := ll[y]$
      $isRoot := False$
  if $(isRoot)$
    let $C = popUntil(s, x)$
    if $(A \cap C \neq \emptyset)$ $r := (A \subseteq C)$
  return r

Algorithm 5: Checking *group* formulas in a quasi-discrete closure space.

Example 7.1. Finite, quasi-discrete models can be encoded as graphs. By this, the CSLCS model checker is able to load also examples with a complex specification. In Figure 16 we show a picture coming from the analysis of the model of Section 5 (above), and the output of the tool (below), colouring in red the nodes that satisfy ϕ_5 from Example 5.4. In particular, the model is based on a discrete version of a vectorial illustration. Execution times for checking the formulas presented in Section 5 depend, indeed, on the resolution of the discrete image. Even though we do not aim at providing benchmarks in this work, just as a hint on execution times, we remark that when the number of points in the image is around one million, verification of formula ϕ_5 takes around two seconds on a standard (at the time of writing) laptop with 8 gigabytes of main memory.

Example 7.2. In Figure 17 we provide a small, black and white image. For A an arbitrary set of points, consider the informal statement "A is located in a *white* area, and it is *collectively surrounded* by a *black* area". The intuition here is that all the points of A should

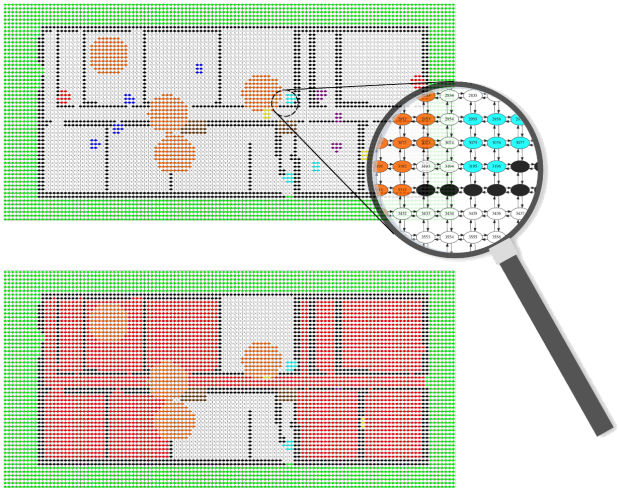

Figure 16: The model of Section 5 rendered as a graph, and the result of model checking formula ϕ_5 of Example 5.4.

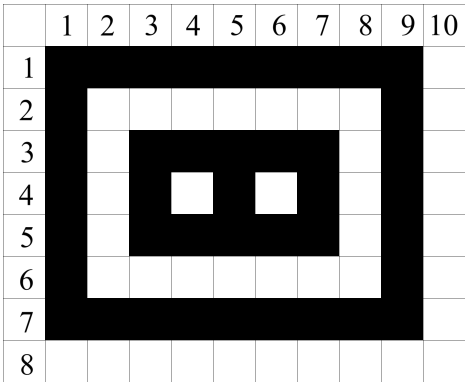

Figure 17: A test case for the property of being *collectively surrounded* (see Example 7.2).

be immersed in *the same* white area. However, the meaning of *same* is not thoroughly specified. A very liberal interpretation of *collectively surrounded* could let the formula be true at any set A such that all points in A individually satisfy $\mathit{white}\,\mathcal{S}\,\mathit{black}$. This is expressed by the CSLCS formula $\forall(\mathit{white}\,\mathcal{S}\,\mathit{black})$. Here "the same area" means "the same subset". This notion can be refined. For example, in Figure 17, let $A = \{(4,4),(6,4)\}$ and $B = \{(4,6),(6,6)\}$ (the first coordinate is the horizontal one). It is also sensible to let "collectively surrounded" tell A and B apart, as B lays in a *connected* white area surrounded by black points, whereas A does not enjoy such property. In this case, "the same area" is defined as "the same connected white area". The derived connective $\mathcal{CS}$ from Definition 4.6 is designed to do this.

The CSLCS model checker can be used to verify these two properties on some subsets of the space. First, we verify whether points at coordinates $(4,4),(4,6),(6,4),(6,6)$ individually satisfy $\mathit{white}\,\mathcal{S}\,\mathit{black}$. This is checked by expanding the definition of the $\forall$ connective, from Definition 4.3. Indeed, the model checker answers *true* to this query. The next step is to tell

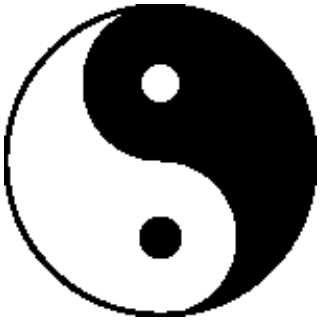

Figure 18: This image is not partitioned by properties *black* and *white*.

Figure 19: This image is partitioned by properties *black* and *white*.

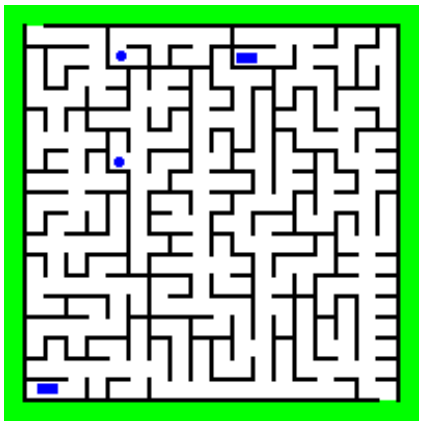

Figure 20: A maze.

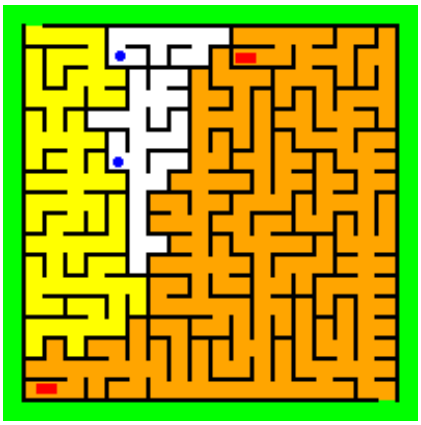

Figure 21: Model checker output.

apart different sets of points using the definition of the $\mathcal{CS}$ connective. The definition of *black* $\mathcal{CS}$ *white* is checked on three different sets. The answer is *true* on sets $\{(4,4)\}$ and $\{(4,6),(6,6)\}$ and *false* on the sets $\{(4,4),(6,4)\}$ and $\{(4,4),(4,6)\}$.

Example 7.3. Using CSLCS it is possible to check whether a given space is *partitioned*, that is, each atomic property lies in a separate area of the image without mixing. We provided a formal definition of such property in Definition 4.6, by the means of the $\mathcal{CP}$ connective. For example, we consider two digital images having only black and white pixels. The CSLCS model checker returns *false* on Figure 18, and *true* on Figure 19, when requested to verify that $\mathit{white}\,\mathcal{CP}\,\mathit{black}$ is globally satisfied according to Definition 4.5.

Example 7.4. In Figure 20 we present another example of how SLCS can be used for classifying points in a digital image. We use a digital image representing a maze. The green area is the exit. The blue areas are starting points. Three formulas are used to identify interesting areas. Such formulas implicitly make use of the surrounded operator, by the means of the derived operators $\mathcal{R}$ and $\mathcal{T}$ (see Section 3).

$$\mathit{toExit} = \mathit{white}\,\mathcal{T}\,\mathit{green}$$
$$\mathit{fromStartToExit} = \mathit{toExit} \wedge (\mathit{white}\,\mathcal{T}\,\mathit{blue})$$
$$\mathit{startCanExit} = \mathit{blue}\,\mathcal{T}\,\mathit{fromStartToExit}$$

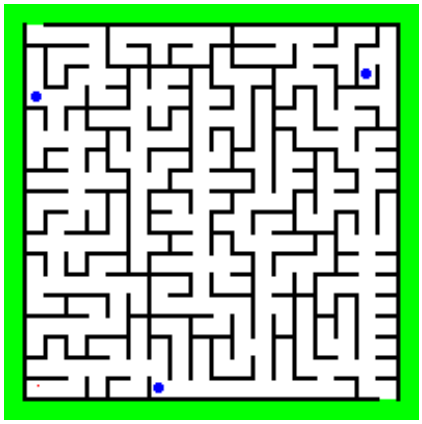

Figure 22: Blue circles are not able to reach the *same* exit.

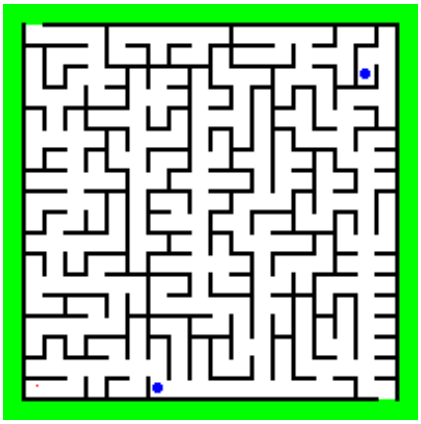

Figure 23: Blue circles are able to reach the *same* exit.

The output of the tool is in Figure 21. The red colour denotes points satisfying *startCanExit*, that is, starting points from which the exit can be reached (for the sake of readability, we have depicted these areas in a rectangular shape, but the tool is obviously not aware of the difference in shape). Orange and yellow indicate the two regions through which the exit can be reached (formula *toExit*). The orange region includes moreover a start point (formula *fromStartToExit*).

Example 7.5. We continue from Example 7.4 to show how collective formulas can easily distinguish models having similar individual properties. In Figure 22, the three blue circles in the maze can all reach the exit; however, they cannot "collectively" do so, as they cannot join and get out through the same exit. In Figure 23, on the other hand, the blue circles *can* get out through the same exit. Importing definitions from Example 7.4, the model checker is able to tell the difference between these two models. When invoked on the formula $blue \prec (\mathcal{G}((blue \vee white)\, \mathcal{T}\, green))$, the tool returns $false$ in the first model, and $true$ in the second one. The given formula, which is interpreted globally (in the sense of Definition 4.5), asserts that all the blue points are part of a strongly connected component of points that can reach the (green) exit passing by points that are either blue or white.

Example 7.6. In Figure 24 we show a digital image[9] depicting a portion of the map of Pisa, featuring a red circle which denotes a train station. Streets of different importance are painted with different colours in the map. The CSLCS model checker is used to identify and colour the area surrounding the station which is delimited by main streets, including the delimiting main streets. The output of the tool is shown in Figure 25, where the station area is coloured in orange, the surrounding main streets are red, and other main streets are in green.

8. Conclusions and Future Work

Spatial logics have been studied extensively in the past as a spatial interpretation of modal logics [APHvB07], with particular emphasis on descriptive languages and aspects such as

[9] © *OpenStreetMap contributors* – `http://www.openstreetmap.org/copyright`.

Figure 24: Input: the map of a town.

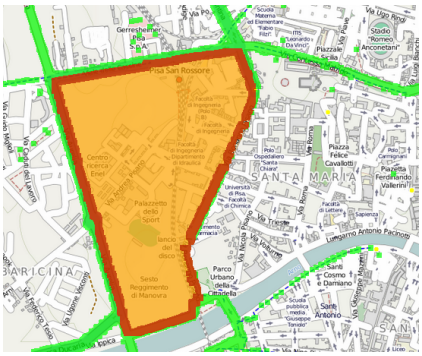

Figure 25: Output of the tool.

completeness, decidability, complexity, that are very relevant for mathematical logics. In this paper we have developed this approach in a different direction, namely that of formal and automatic verification and in particular that of spatial model-checking. This focus required us to take several constraints into consideration. On one hand, our aim was to remain as general as possible, in such a way that the developed spatial model checking algorithms can be applied on a wide variety of spatial representations, including forms of continuous space, discrete space, directed and undirected graphs, possibly extended with metric spaces. On the other hand, efficient and effective model checking procedures require finite structures. To this purpose the theoretical framework of closure spaces (a generalisation of topological spaces) has been explored. This framework provides a set of useful basic abstract spatial operators (closure, interior, boundary and many derived ones) that provide a structured way to define higher level spatial logic operators. Moreover, we have shown that they are also suitable for the development of efficient spatial model checking algorithms in which these same closure space based operators play a role as well.

In particular, in [CLLM14a] we have defined the spatial logic SLCS, stemming from the tradition of topological interpretations of modal logics, dating back to earlier logicians such as Tarski, where modalities describe neighbourhood. The topological definitions have been lifted to a more general setting, also encompassing discrete, graph-based structures. In the present paper an alternative, path-based, definition of the logic has been provided which is more general than that presented in [CLLM14a] and is shown to coincide with the latter in the case of quasi-discrete closure spaces. In addition, the framework has been extended with the *propagation* operator. This operator captures the notion of spatial propagation; intuitively the formula $\phi \mathcal{P} \psi$ describes a situation in which the points satisfying ψ can be reached by paths rooted in points satisfying ϕ and, for the rest, composed only of points satisfying ψ.

Furthermore, we have introduced a *collective* logic, which borrows from the spatial logics tradition, but introduces properties that characterise "collective", spatial features of *sets*, rather than individuals. For both logics, an efficient model-checking algorithm has been defined and implemented, operating on finite, quasi-discrete closure models.

Future work aims at considering temporal reasoning in addition to spatial verification in order to address system evolution and dynamics within a single logic. Both the theoretical nature of this problem, and the efficiency of model checking algorithms, should be investigated. In [KKWZ07], "snapshot" models are considered, consisting of a temporal model (e.g., a

Kripke frame) where each state is in turn a spatial model, and atomic formulas of the temporal fragment are replaced by spatial formulas. The various possible combinations of temporal and spatial operators, in linear and branching time, are examined therein, for the case of topological models, and basic modal formulas. First results on the extension of snapshot models based on *closure* spaces, and the study of spatio-temporal surrounded operators, has led to an extension of SLCS with the branching time logic CTL (Computation Tree Logic [CE82]) and is presented in [CGL$^+$15, CLMP15, CLLM16]. It provides spatio-temporal reasoning and model checking. However, the automated verification of snapshot models is susceptible to state-space explosion problems as spatial formulas need to be recomputed at every state. We will therefore also study how to exploit the fact that changes of space over time are typically incremental and local in nature. Metrics and distance functions can be added in an orthogonal way providing further spatial richness. The theoretical approach pursued in the present paper is starting to find its way to applications such as the detection and analysis of emergent spatial patterns [NBC$^+$15] in behaviour modelled as reaction-diffusion equations, such as those involved in the emergence of patterns in animal fur first studied by Turing. In [NBC$^+$15] the closure space based model checking algorithms have been extended with metric spaces and signal temporal logic leading to monitoring algorithms for a linear time spatio-temporal logic. The logic has qualitative and quantitative semantics, and monitoring algorithms have been designed and implemented. It can be used to verify interesting spatial-temporal properties such as the robustness of patterns to perturbations. Other ongoing applications of spatio-temporal model checking are the analysis of emergent spatio-temporal phenomena, such as the phenomenon of *clumping* (that is, buses with too short headway) in public urban bus transportation systems [CGL$^+$14] and the formation of spatial clusters of full stations in bike sharing systems [CLMP15]. The latter has been also analysed in [CLM$^+$16], where statistical spatio-temporal model checking has been used to infer quantitative information like, for example, the probability of cluster formation. In a completely different domain, preliminary work showed very interesting results, combining spatial model checking with *texture analysis* to segment tumour and oedema in medical images, which is of immediate relevance for automatic contouring applications used in radiotherapy [BCLM16].

Further promising ideas are presented both in [Gal03], where principles of "continuous change" are proposed in the setting of closure spaces, and in [KM07] where spatio-temporal models are generated by locally-scoped update functions, in order to describe dynamic systems. Another interesting alternative approach to describe spatial properties is based on so-called quad trees. Such trees are constructed by recursively partitioning images into quadrants. The spatio-temporal logic SpaTeL [HJK$^+$15] is based on such a spatial superposition logic.

In the setting of collective adaptive systems, it would be relevant to extend the basic framework we presented here with aspects related to distances or metrics (e.g., distance-bounded variants of the surrounded and propagation operators) and probabilistic aspects, using, e.g., atomic propositions that are probability distributions. In this work we have considered connectedness and related properties as the most basic forms of collective properties. Indeed, such properties give the logic a "global" flavour, witnessed by our Definition 4.5. In this respect, CSLCS is similar to $\mathcal{S}4_u$, even though connectedness can not be represented in the latter. Connectedness may be added as a predicate to spatial logics (see [KPWZ10]). An in-depth comparison between $\mathcal{S}4_u$, spatial logics with connectedness, and CSLCS will be considered in future work, possibly taking into account the work of [Sla09]

on connectedness in closure spaces. Other logics that consider sets of points rather than properties of individual points are those based on adjacency spaces such as the region calculus studied in discrete mereotopology [Gal99, Gal14]. In the context of collective properties, one could also consider arbitrary nesting of collective and individual formulas; however, such enhancements translate to inefficient algorithms in the classical exhaustive model checking procedures, as one should enumerate all subsets of the considered set of points. In order to overcome such issues, a *symbolic model checking* approach could be used to represent solution sets without explicit enumeration.

A further challenge in spatial and spatio-temporal reasoning is posed by recursive spatial formulas, *a la* μ-calculus, especially on infinite structures with relatively straightforward generating functions (think of fractals, or fluid flow analysis of continuous structures). Such infinite structures could be described by topologically enhanced variants of ω-automata; more generally speaking, the automata-theoretic approach to logics and verification is certainly of interest also in the field of spatial logics. Classes of automata exist living in specific topological structures; an example is given by *nominal automata* (see e.g., [BKL11, GC11, KST12]), that can be defined using presheaf toposes [FS06], although retaining finite, computationally efficient representations [CM10]. This standpoint could be enhanced with notions of neighbourhood coming from closure spaces, with the aim of developing a unifying theory of languages and automata describing physical spaces, graphs, and process calculi with resources. Finally, a more profound study of the generalisation of the notion of paths in closure spaces could lead to further interesting theoretical results.

9. Acknowledgements

This research has been partially funded by the EU FET Proactive project QUANTICOL (nr. 600708). The authors also wish to thank the anonymous reviewers for their valuable comments and suggestions.

References

[Aie02] Marco Aiello. *Spatial Reasoning: Theory and Practice*. PhD thesis, Institute of Logic, Language and Computation, University of Amsterdam, 2002.

[APHvB07] Marco Aiello, Ian Pratt-Hartmann, and Johan van Benthem, editors. *Handbook of Spatial Logics*. Springer, 2007.

[AS15] Francesco Luca De Angelis and Giovanna Di Marzo Serugendo. A logic language for run time assessment of spatial properties in self-organizing systems. In *2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASO Workshops 2015, Cambridge, MA, USA, September 21-25, 2015*, pages 86–91. IEEE Computer Society, 2015.

[BB07] Johan van Benthem and Guram Bezhanishvili. Modal logics of space. In Aiello et al. [APHvB07], pages 217–298.

[BCLM16] Gina Belmonte, Vincenzo Ciancia, Diego Latella, and Mieke Massink. From collective adaptive systems to human centric computation and back: Spatial model checking for medical imaging. In Maurice H. ter Beek and Michele Loreti, editors, *Proceedings of the Workshop on FORmal methods for the quantitative Evaluation of Collective Adaptive SysTems, FORECAST@STAF 2016, Vienna, Austria, 8 July 2016.*, volume 217 of *EPTCS*, pages 81–92, 2016.

[BHLM13] Luca Bortolussi, Jane Hillston, Diego Latella, and Mieke Massink. Continuous approximation of collective system behaviour: A tutorial. *Performance Evaluation*, 70(5):317 – 349, 2013.

[BHMU11] Arne T. Bittig, Fiete Haack, Carsten Maus, and Adelinde M. Uhrmacher. Adapting rule-based model descriptions for simulating in continuous and hybrid space. In François Fages, editor,

Computational Methods in Systems Biology, 9th International Conference, CMSB 2011, Paris, France, September 21-23, 2011. Proceedings, pages 161–170. ACM, 2011.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[BKL11] Mikołaj Bojańczyk, Bartek Klin, and Slawomir Lasota. Automata with group actions. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 355–364. IEEE Computer Society, 2011.

[CC03] L. Caires and L. Cardelli. A spatial logic for concurrency (part I). *Information and Computation*, 186(2):194–235, 2003.

[CE82] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Dexter Kozen, editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Berlin Heidelberg, 1982.

[CG00] Luca Cardelli and Andrew D. Gordon. Anytime, anywhere: Modal logics for mobile ambients. In *Proceedings of the 30th SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'00)*, pages 365–377, 2000.

[CG12] Luca Cardelli and Philippa Gardner. Processes in space. *Theor. Comput. Sci.*, 431:40–55, 2012.

[CGG02] Luca Cardelli, Philippa Gardner, and Giorgio Ghelli. A spatial logic for querying graphs. In Peter Widmayer, Francisco Triguero Ruiz, Rafael Morales Bueno, Matthew Hennessy, Stephan Eidenbenz, and Ricardo Conejo, editors, *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, volume 2380 of *Lecture Notes in Computer Science*, pages 597–610. Springer, 2002.

[CGL+14] Vincenzo Ciancia, Stephen Gilmore, Diego Latella, Michele Loreti, and Mieke Massink. 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*, pages 32–37. IEEE Computer Society, 2014.

[CGL+15] Vincenzo Ciancia, Gianluca Grilletti, Diego Latella, Michele Loreti, and Mieke Massink. An experimental spatio-temporal model checker. In *Software Engineering and Formal Methods - SEFM 2015 Collocated Workshops: ATSE, HOFM, MoKMaSD, and VERY*SCART, York, UK, September 7-8, 2015, Revised Selected Papers*, volume 9509 of *Lecture Notes in Computer Science*, pages 297–311. Springer, 2015.

[CLBR09] Augustin Chaintreau, Jean-Yves Le Boudec, and Nikodin Ristanovic. The age of gossip: Spatial mean field regime. In *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, SIGMETRICS '09, pages 109–120, New York, NY, USA, 2009. ACM.

[CLLM14a] V. Ciancia, D. Latella, M. Loreti, and M. Massink. Specifying and Verifying Properties of Space. In Springer, editor, *The 8th IFIP International Conference on Theoretical Computer Science, TCS 2014, Track B*, volume 8705 of *Lecture Notes in Computer Science*, pages 222–235, 2014.

[CLLM14b] Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Specifying and verifying properties of space. Technical Report TR-QC-06-2014, QUANTICOL, 2014.

[CLLM16] Vincenzo Ciancia, Diego Latella, Michele Loreti, and Mieke Massink. Spatial logic and spatial model checking for closure spaces. In Marco Bernardo, Rocco De Nicola, and Jane Hillston, editors, *Formal Methods for the Quantitative Evaluation of Collective Adaptive Systems - 16th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2016, Bertinoro, Italy, June 20-24, 2016, Advanced Lectures*, volume 9700 of *Lecture Notes in Computer Science*, pages 156–201. Springer, 2016.

[CLM+16] Vincenzo Ciancia, Diego Latella, Mieke Massink, Rytis Paškauskas, and Andrea Vandin. A tool-chain for statistical spatio-temporal model checking of bike sharing systems. In Tiziana Margaria and Bernhard Steffen, editors, *7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 5-14, 2016, Proceedings*, volume 9952 of *Lecture Notes in Computer Science*. Springer, 2016.

[CLMP15] Vincenzo Ciancia, Diego Latella, Mieke Massink, and Rytis Pakauskas. Exploring spatio-temporal properties of bike-sharing systems. In *2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASO Workshops 2015, Cambridge, MA, USA, September 21-25, 2015*, pages 74–79. IEEE Computer Society, 2015.

[CM10] Vincenzo Ciancia and Ugo Montanari. Symmetries, local names and dynamic (de)-allocation of names. *Inf. Comput.*, 208(12):1349 – 1367, 2010.

[DBVZ95] Alberto Del Bimbo, Enrico Vicario, and Daniele Zingoni. Symbolic description and visual querying of image sequences using spatio-temporal logic. *IEEE Trans. Knowl. Data Eng.*, 7(4):609–622, 1995.

[DFP98] Rocco De Nicola, Gian Luigi Ferrari, and Rosario Pugliese. Klaim: A kernel language for agents interaction and mobility. *IEEE Trans. Software Eng.*, 24(5):315–330, 1998.

[FS06] Marcelo P. Fiore and Sam Staton. Comparing operational models of name-passing process calculi. *Inf. Comput.*, 204(4):524–560, 2006.

[Gal99] Antony Galton. The mereotopology of discrete space. In Christian Freksa and DavidM. Mark, editors, *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*, volume 1661 of *Lecture Notes in Computer Science*, pages 251–266. Springer Berlin Heidelberg, 1999.

[Gal03] Antony Galton. A generalized topological view of motion in discrete space. *Theoretical Computer Science*, 305(1–3):111 – 134, 2003.

[Gal14] Antony Galton. Discrete mereotopology. In Claudio Calosi and Pierluigi Graziani, editors, *Mereology and the Sciences*, pages 293–321. Springer International Publishing, 2014.

[GBB14] Ebru Aydin Gol, Ezio Bartocci, and Calin Belta. A formal methods approach to pattern synthesis in reaction diffusion systems. In *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*, pages 108–113. IEEE, 2014.

[GBC+08] R. Grosu, E. Bartocci, F. Corradini, E. Entcheva, S. A. Smolka, and A. Wasilewska. Learning and detecting emergent behavior in networks of cardiac myocytes. In *Proc. of HSCC 2008, the 11th ACM international conference on Hybrid Systems: Computation and Control*, 2008.

[GC11] Murdoch James Gabbay and Vincenzo Ciancia. Freshness and name-restriction in sets of traces with names. In Martin Hofmann, editor, *Foundations of Software Science and Computational Structures - 14th International Conference, FOSSACS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, volume 6604 of *Lecture Notes in Computer Science*, pages 365–380. Springer, 2011.

[GL07] Fabio Gadducci and Alberto Lluch-Lafuente. Graphical encoding of a spatial logic for the *pi* -calculus. In Till Mossakowski, Ugo Montanari, and Magne Haveraaen, editors, *Algebra and Coalgebra in Computer Science, Second International Conference, CALCO 2007, Bergen, Norway, August 20-24, 2007, Proceedings*, volume 4624 of *Lecture Notes in Computer Science*, pages 209–225. Springer, 2007.

[Gra09] Marco Grandis. *Directed algebraic topology : models of non-reversible worlds*. Cambridge University Press, Cambridge, 2009.

[GSC+09] Radu Grosu, Scott A. Smolka, Flavio Corradini, Anita Wasilewska, Emilia Entcheva, and Ezio Bartocci. Learning and detecting emergent behavior in networks of cardiac myocytes. *Commun. ACM*, 52(3):97–105, March 2009.

[HJK+15] I. Haghighi, A. Jones, J. Z. Kong, E. Bartocci, Grosu R., and C. Belta. SpaTeL: A Novel Spatial-Temporal Logic and Its Applications to Networked Systems. In *Proc. of HSCC*, 2015.

[JEU08] Mathias John, Roland Ewald, and Adelinde M. Uhrmacher. A spatial extension to the pi calculus. *Electr. Notes Theor. Comput. Sci.*, 194(3):133–148, 2008.

[KKWZ07] Roman Kontchakov, Agi Kurucz, Frank Wolter, and Michael Zakharyaschev. Spatial logic + temporal logic = ? In Aiello et al. [APHvB07], pages 497–564.

[KM07] Philip Kremer and Grigori Mints. Dynamic topological logic. In Aiello et al. [APHvB07], pages 565–606.

[Kov08] V. A. Kovalevsky. *Geometry of Locally Finite Spaces: Computer Agreeable Topology and Algorithms for Computer Imagery*. Editing House Dr. Baerbel Kovalevski, 2008.

[KPWZ10] Roman Kontchakov, Ian Pratt-Hartmann, Frank Wolter, and Michael Zakharyaschev. Spatial logics with connectedness predicates. *Logical Methods in Computer Science*, 6(3), 2010.

[KR89] T. Yung Kong and Azriel Rosenfeld. Digital topology: Introduction and survey. *Computer Vision, Graphics, and Image Processing*, 48(3):357–393, 1989.

[KST12] Alexander Kurz, Tomoyuki Suzuki, and Emilio Tuosto. On nominal regular languages with binders. In Lars Birkedal, editor, *Foundations of Software Science and Computational Structures*

- *15th International Conference, FOSSACS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*, volume 7213 of *Lecture Notes in Computer Science*, pages 255–269. Springer, 2012.

[Mil09] Robin Milner. *The Space and Motion of Communicating Agents*. Cambridge University Press, 2009.

[NB14] Laura Nenzi and Luca Bortolussi. Specifying and monitoring properties of stochastic spatio-temporal systems in signal temporal logic. In Moshe Haviv, William J. Knottenbelt, Lorenzo Maggi, and Daniele Miorandi, editors, *8th International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2014, Bratislava, Slovakia, December 9-11, 2014*. ICST, 2014.

[NBC+15] Laura Nenzi, Luca Bortolussi, Vincenzo Ciancia, Michele Loreti, and Mieke Massink. Qualitative and quantitative monitoring of spatio-temporal properties. In Ezio Bartocci and Rupak Majumdar, editors, *Runtime Verification - 6th International Conference, RV 2015 Vienna, Austria, September 22-25, 2015. Proceedings*, volume 9333 of *Lecture Notes in Computer Science*, pages 21–37. Springer, 2015.

[Ros79] Azriel Rosenfeld. Digital topology. *The American Mathematical Monthly*, 86(8):621–630, 1979.

[RS85] John Reif and A.P. Sistla. A multiprocess network logic with temporal and spatial modalities. *Journal of Computer and System Sciences*, 30(1):41 – 53, 1985.

[Sla09] Josef Slapal. Another approach to connectedness with respect to a closure operator. *Applied Categorical Structures*, 17(6):603–612, 2009.

[SW07] Michael B. Smyth and Julian Webster. Discrete spatial models. In Springer [APHvB07], pages 713–798.

[Tar72] Robert Tarjan. Depth first search and linear graph algorithms. *SIAM Journal on Computing*, 1972.

[TPGN15] Christos Tsigkanos, Liliana Pasquale, Carlo Ghezzi, and Bashar Nuseibeh. Ariadne: Topology aware adaptive security for cyber-physical systems. In *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2*, pages 729–732. IEEE, 2015.

Appendix A. Proofs

The proofs of Proposition 2.10, Proposition 2.12, and Proposition 2.14 are straightforward, and have been omitted from this paper. Full proofs are available in [CLLM14b].

Proof. (of Proposition 2.17)

Axiom 1:

$$\mathcal{C}_R(\emptyset) = \emptyset \cup \{x \in X \mid \exists a \in \emptyset.(a, x) \in R\} = \emptyset$$

Axiom 2:

$$\begin{array}{ll} & A \\ \subseteq & [\, A \subseteq A \cup B \,] \\ & \mathcal{C}_R(A) \end{array}$$

Axiom 3:

$$
\begin{aligned}
& \mathcal{C}_R(A \cup B) \\
= \quad & A \cup B \cup \{x \in X \mid \exists c \in A \cup B.(c,x) \in R\} \\
= \quad & [\, c \in A \cup B \iff c \in A \vee c \in B \,] \\
& A \cup B \cup \{x \in X \mid \exists c \in A.(c,x) \in R\} \cup \{x \in X \mid \exists c \in B.(c,x) \in R\} \\
= \quad & \mathcal{C}_R(A) \cup \mathcal{C}_R(B)
\end{aligned}
$$

□

Proof. (of Proposition 2.20)
Equation (2.8):

$$
\begin{aligned}
& \mathcal{I}(A) \\
= \quad & \overline{\mathcal{C}_R(\overline{A})} \\
= \quad & \overline{\overline{A} \cup \{x \in X \mid \exists a \in \overline{A}.(a,x) \in R\}} \\
= \quad & A \cap \{x \in X \mid \neg\exists a \in \overline{A}.(a,x) \in R\} \\
= \quad & \{x \in A \mid \neg\exists a \in \overline{A}.(a,x) \in R\}
\end{aligned}
$$

Equation (2.9):

$$
\begin{aligned}
& \mathcal{B}^-(A) \\
= \quad & A \setminus \mathcal{I}(A) \\
= \quad & A \setminus \{x \in A \mid \neg\exists a \in \overline{A}.(a,x) \in R\} \\
= \quad & A \cap \{x \in A \mid \exists a \in \overline{A}.(a,x) \in R\} \\
= \quad & \{x \in A \mid \exists a \in \overline{A}.(a,x) \in R\}
\end{aligned}
$$

Equation (2.10):

$$
\begin{aligned}
& \mathcal{B}^+(A) \\
= \quad & \mathcal{C}(A) \setminus A \\
= \quad & (A \cup \{x \in X \mid \exists a \in A.(a,x) \in R\}) \setminus A \\
= \quad & (A \cup \{x \in X \mid \exists a \in A.(a,x) \in R\}) \cap \overline{A} \\
= \quad & (A \cap \overline{A}) \cup (\{x \in X \mid \exists a \in A.(a,x) \in R\} \cap \overline{A}) \\
= \quad & \{x \in \overline{A} \mid \exists a \in A.(a,x) \in R\}
\end{aligned}
$$

□

Proof. (of Lemma 2.33) For one direction of the proof, assume p is a continuous function. Importing definitions from Definition 2.31 and the statement of Lemma 2.33, we have

$$
\begin{aligned}
& (i, i+1) \in \mathit{Succ} \\
\Longrightarrow\ & i+1 \in \mathcal{C}_{\mathit{Succ}}(\{i\}) \\
\Longrightarrow\ & [\, p \text{ continuous} \,] \\
& p(i+1) \in \mathcal{C}_R(p(\{i\})) \\
\Longleftrightarrow\ & p(i+1) \in \mathcal{C}_R(\{p(i)\}) \\
\Longleftrightarrow\ & p(i+1) \in \{p(i)\} \cup \{x \mid (p(i), x) \in R\} \\
\Longleftrightarrow\ & p(i+1) = p(i) \vee (p(i), p(i+1)) \in R
\end{aligned}
$$

For the other direction, given a path $x_{i \in I}$ R, define $p(i) = x_i$. Continuity of p is straightforward. □

Proof. (of Proposition 3.4)

(1) $\mathcal{M}, x \models \phi_1 \,\mathcal{R}\, \phi_2$

$$
\begin{aligned}
\Longleftrightarrow\ & [\,\text{Definition of } \mathcal{R}\,] \\
& \mathcal{M}, x \models \neg(\neg\phi_2 \,\mathcal{S}\, \neg\phi_1) \\
\Longleftrightarrow\ & [\,\text{Definition of } \mathcal{S}\,] \\
& \neg(\mathcal{M}, x \models \neg\phi_2 \text{ and} \\
& \forall p : x \rightsquigarrow \infty \forall l \in \mathbb{N} : \mathcal{M}, p(l) \models \neg\neg\phi_2 \Rightarrow \exists k \in \{1, \ldots, l\} : \mathcal{M}, p(k) \models \neg\phi_1) \\
\Longleftrightarrow\ & \neg(\mathcal{M}, x \models \neg\phi_2 \text{ and} \\
& \forall p : x \rightsquigarrow \infty \forall l \in \mathbb{N} :\ \neg(\mathcal{M}, p(l) \models \phi_2) \vee (\exists k \in \{1, \ldots, l\} : \mathcal{M}, p(k) \models \neg\phi_1)) \\
\Longleftrightarrow\ & \mathcal{M}, x \models \phi_2 \text{ or} \\
& \exists p : x \rightsquigarrow \infty \exists l \in \mathbb{N} :\ \mathcal{M}, p(l) \models \phi_2 \wedge \neg(\exists k \in \{1, \ldots, l\} : \mathcal{M}, p(k) \models \neg\phi_1) \\
\Longleftrightarrow\ & \exists p : x \rightsquigarrow \infty \exists l \in \mathbb{N} :\ \mathcal{M}, p(l) \models \phi_2 \wedge \forall k \in \{1, \ldots, l\} : \mathcal{M}, p(k) \models \phi_1
\end{aligned}
$$

(2) $\mathcal{M}, x \models \phi_1 \mathcal{A} \phi_2$

$\iff$ [Definition of $\mathcal{A}$]

$\mathcal{M}, x \models \neg(\phi_1 \mathcal{P} \neg\phi_2)$

$\iff$ [Definition of $\mathcal{P}$]

$\neg(\mathcal{M}, x \models \neg\phi_2$ and

$\exists y : \mathcal{M}, y \models \phi_1$ and $\exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \forall i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \neg\phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\neg(\exists y : \mathcal{M}, y \models \phi_1$ and $\exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \forall i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \neg\phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\forall y : \neg(\mathcal{M}, y \models \phi_1$ and $\exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \forall i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \neg\phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\forall y : \neg\mathcal{M}, y \models \phi_1$ or $\neg(\exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \forall i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \neg\phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\forall y : \neg\mathcal{M}, y \models \phi_1$ or $\forall p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \neg(\forall i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \neg\phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\forall y : \neg\mathcal{M}, y \models \phi_1$ or $\forall p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \exists i \in \{1, \ldots, l-1\} : \neg\mathcal{M}, p(i) \models \neg\phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\forall y : \neg\mathcal{M}, y \models \phi_1$ or $\forall p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \exists i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\forall y : \mathcal{M}, y \models \phi_1 \Rightarrow \forall p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \exists i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \phi_2)$

$\iff$ $\mathcal{M}, x \models \phi_2$ or

$\forall y : \mathcal{M}, y \models \phi_1 : \forall p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty : \exists i \in \{1, \ldots, l-1\} : \mathcal{M}, p(i) \models \phi_2)$

(3) $\mathcal{M}, x \models \mathcal{E} \phi$

$\iff$ [Definition of $\mathcal{E}$]

$\mathcal{M}, x \models \phi \mathcal{S} \bot$

$\iff$ [Definition of $\mathcal{S}$]

$\forall p : x \rightsquigarrow \infty \forall l \in \mathbb{N} : \mathcal{M}, p(l) \models \neg\phi \Rightarrow \exists k \in \{1, \ldots, l\} : \mathcal{M}, p(k) \models \bot$

$\iff$ $\forall p : x \rightsquigarrow \infty \forall l \in \mathbb{N} : \mathcal{M}, p(l) \models \phi$

(4) $\mathcal{M}, x \models \mathcal{F}\phi$

$\iff$ [Definition of $\mathcal{F}$]

$\mathcal{M}, x \models \neg\, \mathcal{E}\, \neg\phi$

$\iff$ [Proposition 3.4 (3)]

$\neg(\forall p : x \rightsquigarrow \infty \forall l \in \mathbb{N} : \mathcal{M}, p(l) \models \neg\phi)$

$\iff \exists p : x \rightsquigarrow \infty \exists l \in \mathbb{N} : \mathcal{M}, p(l) \models \phi$ □

Proof. (of Theorem 3.7) Consider a quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}), \mathcal{V})$ and suppose $\mathcal{M}, x \models \phi_1 \mathcal{U} \phi_2$ as defined in [CLLM14a], that is, suppose there is a set A with $x \in A$, $\forall y \in A. \mathcal{M}, y \models \phi_1$, and $\forall z \in \mathcal{B}^+(A). \mathcal{M}, z \models \phi_2$. Let p be a $\mathfrak{N}$-path, with $p : x \rightsquigarrow \infty$, and let l be such that $\mathcal{M}, p(l) \models \neg\phi_1$. Consider the set $K^- = \{k \mid \forall h \in \{0, \ldots, k\}. p(h) \in A\}$. Since $0 \in K^-$, we have $K^- \neq \emptyset$. Consider the complement of K^-, namely $K^+ = \mathbb{N} \setminus K^-$. Since all points in A satisfy ϕ_1, and $p(l) \models \neg\phi_1$, we have $l \in K^+$, thus $K^+ \neq \emptyset$. By existence of l, K^- is finite, thus, being non-empty, it has a greatest element. Being a non-empty subset of the natural numbers, K^+ has a least element. Let $k^- = \max K^-$ and $k^+ = \min K^+$. By definition of K^-, if $k \in K^-$ and $h \in [0, k)$, then $h \in K^-$. In particular, for all $h \leq k^-$, we have $h \in K^-$. By definition of k^-, we have $k^- + 1 \notin K^-$, that is, $k^- + 1 \in K^+$. Therefore, we have $k^- + 1 = k^+$, thus $(k^-, k^+) \in \mathit{Succ}$. Let $S = \{p(k) | k \in K^-\} \subseteq A$. By monotonicity of closure, we have $\mathcal{C}(S) \subseteq \mathcal{C}(A)$. By definition of $\mathcal{C}_{\mathit{Succ}}$, we have $k^+ \in \mathcal{C}_{\mathit{Succ}}(K^-)$, thus by closure-continuity $p(k^+) \in \mathcal{C}(S)$ and therefore $p(k^+) \in \mathcal{C}(A)$. But it is also true that $p(k^+) \notin A$; if $p(k^+) \in A$, then we would have $k^+ \in K^-$, by definition of K^-. Thus, $p(k^+) \in \mathcal{B}^+(A)$, therefore $p(k^+) \models \phi_2$. Note that in particular $k^+ \neq 0$ as $p(0) = x \in A$, and $k^+ \leq l$ as $l \in K^+$ and $k^+ = \min K^+$.

For the other direction, assume $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$ where $\mathcal{C}_R$ is the closure operator derived by a relation R. Consider point x with $\mathcal{M}, x \models \phi_1$, and assume that for each $p : x \rightsquigarrow \infty$ and l such that $\mathcal{M}, p(l) \models \neg\phi_1$ there is $k \in \{1, \ldots, l\}$ such that $\mathcal{M}, p(k) \models \phi_2$. Define the following set:

$$A_x = \{x\} \cup \{y \in X \mid \exists p : x \rightsquigarrow \infty. \exists l > 0. p(l) = y \wedge \forall k \in \{1, \ldots, l\}. \mathcal{M}, p(k) \models \phi_1 \wedge \neg\phi_2\}$$

We will use A_x as a witness of the existence of a set A, in order to prove that $\mathcal{M}, x \models \phi_1 \mathcal{U} \phi_2$ according to [CLLM14a]. Note that by definition of A_x, $x \in A_x$ and $\forall y \in A_x. \mathcal{M}, p(y) \models \phi_1$. We need to show that $\forall z \in \mathcal{B}^+(A_x). \mathcal{M}, z \models \phi_2$. Consider $z \in \mathcal{B}^+(A_x)$. Since $\mathcal{M}$ is based on a quasi-discrete closure space, by Equation (2.10) in Proposition 2.20, we have $z \in \overline{A_x}$ and there is $y \in A_x$ such that $(y, z) \in R$. Suppose $y = x$. Let p be the path defined by $p(0) = x$, $p(i \neq 0) = z$. If $\mathcal{M}, z \models \phi_1$, suppose $\mathcal{M}, z \not\models \phi_2$; then $z \in A_x$, witnessed by the path p, with $l = 1$; therefore, since $z \in \overline{A_x}$ we have $\mathcal{M}, z \models \phi_2$. If $\mathcal{M}, z \not\models \phi_1$, then noting $p(1) = z$, by hypothesis, there is $k \in \{1, \ldots, 1\}$ with $\mathcal{M}, p(k) \models \phi_2$, that is $\mathcal{M}, z \models \phi_2$. Suppose $y \neq x$. Then there are $p : x \rightsquigarrow \infty$ and $l > 0$ such that $p(l) = y \wedge \forall k \in \{1, \ldots, l\}. \mathcal{M}, p(k) \models \phi_1 \wedge \neg\phi_2$. Define p' by $p'(l') = p(l')$ if $l' \leq l$, and $p'(l') = z$ otherwise. The rest of the proof mimics the case $y = x$. If $\mathcal{M}, z \models \phi_1$, then $\mathcal{M}, z \not\models \phi_2$ implies $z \in A_x$, witnessed by p' and $l' = l + 1$, therefore $\mathcal{M}, z \models \phi_2$. If $\mathcal{M}, z \models \neg\phi_1$, then by hypothesis there must be $k \in \{1, \ldots, l + 1\}$ such that $\mathcal{M}, p'(k) \models \phi_2$. By definition of p', it is not possible that $k \in \{1, \ldots, l\}$, thus $k = l + 1$ and $\mathcal{M}, z \models \phi_2$.

By this argument, we have $\mathcal{M}, x \models \phi_1 \mathcal{S} \phi_2$ using the set A_x to verify the definition of satisfaction. □

Proof. (of Lemma 4.4) The results in item 3 and item 2 easily follow from item 1. For item 1, we have:

$$
\begin{aligned}
& \mathcal{M}, A \models_C \forall \phi \\
\iff & [\text{Def. } \forall] \\
& \mathcal{M}, A \models_C \neg \phi \prec \mathcal{G} \bot \\
\iff & [\text{Def. } \prec] \\
& \mathcal{M}, \{x \in A \mid \mathcal{M}, x \models \neg \phi\} \models \mathcal{G} \bot \\
\iff & [\text{Def. } \mathcal{G}] \\
& \exists B \subseteq X. \{x \in A \mid \mathcal{M}, x \models \neg \phi\} \subseteq B \wedge B \text{ is path connected } \wedge \forall z \in B. \mathcal{M}, z \models \bot \\
\iff & [\forall z \in B. \mathcal{M}, z \models \bot \iff B = \emptyset] \\
& \{x \in A \mid \mathcal{M}, x \models \neg \phi\} \subseteq \emptyset \\
\iff & \forall x \in A. \mathcal{M}, x \models \phi
\end{aligned}
$$

□

□

Proof. (of Lemma 6.2) We prove by induction on the syntax of SLCS formulae that for any quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, and for any formula ϕ function `Sat` terminates in at most $\mathcal{O}(size(\phi) \cdot (|X| + |R|))$ steps.

Base of Induction. If $\phi = \top$ or $\phi = p$ the statement follows directly from the definition of `Sat`. Indeed, in both these cases function `Sat` computes the final result in just 1 step.

Inductive Hypothesis. Let ϕ_1 and ϕ_2 be such that for any quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, function $\texttt{Sat}(\mathcal{M}, \phi_i)$, $i = 1, 2$, terminate in at most $\mathcal{O}(size(\phi_i) \cdot (|X| + |R|))$ steps.

Inductive Step.

$\phi = \neg \phi_1$**:** In this case function `Sat` first recursively computes the set $P = \texttt{Sat}(\mathcal{M}, \phi_1)$, then returns $X - P$. By inductive hypothesis, the calculation of P terminates in at most $\mathcal{O}(size(\phi_1) \cdot (|X| + |R|))$ steps, while to compute $X - P$ we need $\mathcal{O}(|X|)$ steps. Hence, $\texttt{Sat}(\mathcal{M}, \neg \phi_1)$ terminates in at most $\mathcal{O}(size(\phi_1) \cdot (|X| + |R|)) + \mathcal{O}(|X|)$. However:

$$
\begin{aligned}
& \mathcal{O}(size(\phi_1) \cdot (|X| + |R|)) + \mathcal{O}(|X|) \\
\leq \; & \mathcal{O}(size(\phi_1) \cdot (|X| + |R|)) + \mathcal{O}(|X| + |R|) \\
= \; & \mathcal{O}((1 + size(\phi_1)) \cdot (|X| + |R|)) \\
= \; & \mathcal{O}(size(\neg \phi_1) \cdot (|X| + |R|))
\end{aligned}
$$

$\phi = \phi_1 \wedge \phi_2$**:** To compute $P = \texttt{Sat}(\mathcal{M}, \phi_1 \wedge \phi_2)$ function `Sat` first computes $P = \texttt{Sat}(\mathcal{M}, \phi_1)$ and $Q = \texttt{Sat}(\mathcal{M}, \phi_2)$. Then the final result is obtained as $P \cap Q$. Like for the previous case, we have that the statement follows from inductive hypothesis and by using the fact that $P \cap Q$ can be computed in at most $\mathcal{O}(|X|)$.

$\phi = \mathcal{N} \phi_1$**:** In this case function `Sat` first computes, in at most $\mathcal{O}(size(\phi_1) \cdot (|X| + |R|))$ steps, the set $P = \texttt{Sat}(\mathcal{M}, \phi_1)$. Then the final result is obtained as $\mathcal{C}_R(P)$. Note that, to compute $\mathcal{C}_R(P)$ one needs $\mathcal{O}(|X| + |R|)$ steps. According to Definition 2.16, $\mathcal{C}_R(P)$ is

obtained as the union, computable in $\mathcal{O}(|X|)$ steps, of P with $\{x \in X | \exists a \in P.(a,x) \in R\}$. The latter can be computed in $\mathcal{O}(|R|)$ steps. Indeed, we need to consider all the *edges* exiting from P. Hence, $\texttt{Sat}(\mathcal{M}, \mathcal{N}\phi_1)$ terminates in a number of steps that is:

$$\begin{array}{rl} & \mathcal{O}(size(\phi_1) \cdot (|X| + |R|)) + \mathcal{O}(|X|) + \mathcal{O}(|R|) \\ = & \mathcal{O}(size(\phi_1) \cdot (|X| + |R|)) + \mathcal{O}(|X| + |R|) \\ = & \mathcal{O}((1 + size(\phi_1)) \cdot (|X| + |R|)) \\ = & \mathcal{O}(size(\mathcal{N}\phi_1) \cdot (|X| + |R|)) \end{array}$$

$\phi = \phi_1 \mathcal{S} \phi_2$: When $\phi = \phi_1 \mathcal{S} \phi_2$ function `Sat` recursively invokes function `CheckSurr` that first computes the sets $V = \texttt{Sat}(\mathcal{M}, \phi_1)$, $Q = \texttt{Sat}(\mathcal{M}, \phi_2)$ and $T = \mathcal{B}^+(V \cup Q)$. By inductive hypothesis, the computations of V and Q terminate in at most $\mathcal{O}(size(\phi_1) \cdot (|X| + |R|))$ and $\mathcal{O}(size(\phi_2) \cdot (|X| + |R|))$ steps, respectively, while T can be computed in $\mathcal{O}(|X| + |R|)$. After that, the loop at the end of function `CheckSurr` is executed. We can observe that:

- a point x is added to T only one time (i.e. if an element is removed from T, it is never reinserted in T);
- all the points in T are eventually removed from T;
- each *edge* in $\mathcal{M}$ is traversed at most one time.

The first two items, together with the fact that $\mathcal{M}$ is finite, guarantee that the loop terminates. The last item guarantees that the loop terminates in at most $\mathcal{O}(|R|)$ steps[10]. Summing up, the computation of $\texttt{Sat}(\mathcal{M}, \phi_1 \mathcal{S} \phi_2)$ terminates in at most

$$\begin{array}{rl} & \mathcal{O}(size(\phi_1) \cdot (|X| + |R|)) + \mathcal{O}(size(\phi_2) \cdot (|X| + |R|)) \\ & + \mathcal{O}(|X| + |R|) + \mathcal{O}(|R|) \\ = & \mathcal{O}((size(\phi_1) + size(\phi_2)) \cdot (|X| + |R|)) + \mathcal{O}(|X| + |R|) \\ = & \mathcal{O}((1 + size(\phi_1) + size(\phi_2)) \cdot (|X| + |R|)) \\ = & \mathcal{O}(size(\phi_1 \mathcal{S} \phi_2) \cdot (|X| + |R|)) \end{array}$$

$\phi = \phi_1 \mathcal{P} \phi_2$: Similarly to the previous case, when $\phi = \phi_1 \mathcal{P} \phi_2$ function `Sat` recursively invokes function `CheckProp` that first computes the sets $V = \texttt{Sat}(\mathcal{M}, \phi_1)$, $Q = \texttt{Sat}(\mathcal{M}, \phi_2)$ and $T = \mathcal{B}^+(V) \cap Q$. By inductive hypothesis, the computations of V and Q terminate in at most $\mathcal{O}(size(\phi_1) \cdot (|X| + |R|))$ and $\mathcal{O}(size(\phi_2) \cdot (|X| + |R|))$ steps, respectively, while T can be computed in $\mathcal{O}(|X| + |R|)$. After that, the loop at the end of function `CheckProp` is executed. We can observe that:

- a point x is added to T only one time (i.e. if an element is removed from T, it is never reinserted in T);
- all the points in T are eventually removed from T;
- each *edge* in $\mathcal{M}$ is traversed at most one time.

The first two items, together with the fact that $\mathcal{M}$ is finite, guarantee that the loop terminates. The last item guarantees that the loop terminates in at most $\mathcal{O}(|X| + |R|)$ steps. Summing up, the computation of $\texttt{Sat}(\mathcal{M}, \phi_1 \mathcal{P} \phi_2)$ terminates in at most

$$\begin{array}{rl} & \mathcal{O}(size(\phi_1) \cdot (|X| + |R|)) + \mathcal{O}(size(\phi_2) \cdot (|X| + |R|)) \\ & + \mathcal{O}(|X| + |R|) + \mathcal{O}(|X| + |R|) \\ = & \mathcal{O}((size(\phi_1) + size(\phi_2)) \cdot (|X| + |R|)) + \mathcal{O}(|X| + |R|) \\ = & \mathcal{O}((1 + size(\phi_1) + size(\phi_2)) \cdot (|X| + |R|)) \\ = & \mathcal{O}(size(\phi_1 \mathcal{P} \phi_2) \cdot (|X| + |R|)) \end{array}$$

□

[10]Note that this is the complexity for a DFS in a graph

Proof. (of Theorem 6.3) The proof proceeds by induction on the syntax of SLCS formulae.

Base of Induction. If $\phi = \top$ or $\phi = p$ the statement follows directly from the definition of function `Sat` and from Definition 3.3.

Inductive Hypothesis. Let ϕ_1 and ϕ_2 be such that for any finite quasi-discrete closure model $\mathcal{M} = ((X, \mathcal{C}_R), \mathcal{V})$, function $x \in \texttt{Sat}(\mathcal{M}, \phi_i)$ if and only if $\mathcal{M}, x \models \phi_i$, for $i = 1, 2$.

Inductive Step.

$\phi = \neg\phi_1$**:** $x \in \texttt{Sat}(\mathcal{M}, \neg\phi_1)$

$\iff$ [Definition of `Sat`]

$x \notin \texttt{Sat}(\mathcal{M}, \phi_1)$

$\iff$ [Inductive Hypothesis]

$\mathcal{M}, x \not\models \phi_1$

$\iff$ [Definition 3.3]

$\mathcal{M}, x \models \neg\phi_1$

$\phi = \phi_1 \wedge \phi_2$**:** $x \in \texttt{Sat}(\mathcal{M}, \phi_1 \wedge \phi_2)$

$\iff$ [Definition of `Sat`]

$x \in \texttt{Sat}(\mathcal{M}, \phi_1) \cap \texttt{Sat}(\mathcal{M}, \phi_2)$

$\iff$ $x \in \texttt{Sat}(\mathcal{M}, \phi_1)$ and $x \in \texttt{Sat}(\mathcal{M}, \phi_2)$

$\iff$ [Inductive Hypothesis]

$\mathcal{M}, x \models \phi_1$ and $\mathcal{M}, x \models \phi_2$

$\iff$ [Definition 3.3]

$\mathcal{M}, x \models \phi_1 \wedge \phi_2$

$\phi = \mathcal{N}\phi_1$**:** $x \in \texttt{Sat}(\mathcal{N}\phi_1)$

$\iff$ [Definition of `Sat`]

$x \in \mathcal{C}_R(\texttt{Sat}(\mathcal{M}, \phi_1))$

$\iff$ [Definition of $\mathcal{C}_R$]

$\exists A \subseteq \texttt{Sat}(\mathcal{M}, \phi_1) : x \in \mathcal{C}_R(A)$

$\iff$ [Inductive Hypothesis]

$\exists A \subseteq X. \forall y \in A. \mathcal{M}, y, \models \phi_i$ and $x \in \mathcal{C}_R(A)$

$\iff$ [Definition 3.3]

$\mathcal{M}, x \models \mathcal{N}\phi_1$

$\phi = \phi_1 \,\mathcal{S}\, \phi_2$**:** We prove that $x \in \texttt{CheckSurr}(\mathcal{M}, \phi_1, \phi_2)$ if and only if $\mathcal{M}, x \models \phi_1 \,\mathcal{S}\, \phi_2$. Function `CheckSurr` takes as parameters a model $\mathcal{M}$ and two SLCS formulas ϕ_1 and ϕ_2 and computes the set of points in $\mathcal{M}$ satisfying $\phi_1 \,\mathcal{S}\, \phi_2$ by removing from $V = \texttt{Sat}(\mathcal{M}, \phi_1)$ all the *bad* points.

A point is *bad* if it can *reach* a point satisfying $\neg\phi_1$ without passing through a point satisfying ϕ_2. Let $Q = \texttt{Sat}(\mathcal{M}, \phi_2)$ be the set of points in $\mathcal{M}$ satisfying ϕ_2. To identify the

bad points in V the function `CheckSurr` performs a *backward search* from $T = \mathcal{B}^+(V \cup Q)$. Note that any *path exiting* from $V \cup Q$ has to pass through points in T. Moreover, the latter only contains points that satisfy neither ϕ_1 nor ϕ_2, by definition. Until T is empty, function `CheckSurr` first picks all the elements x in T and then removes from V the set of (bad) points N that are in $V - Q$ and that can reach x in *one step*. At the end of each iteration the set T contains the set of *bad* points discovered in the last iteration. The proof proceeds in two steps. The first step guarantees that if x does not satisfy $\phi_1 \mathcal{S} \phi_2$, then x is eventually removed from V. The second step shows that if x is removed from V then x does not satisfy $\phi_1 \mathcal{S} \phi_2$.

Note that, by Inductive Hypothesis, we have that:

$$x \in V = \texttt{Sat}(\mathcal{M}, \phi_1) \Leftrightarrow \mathcal{M}, x \models \phi_1 \tag{A.1}$$

$$x \in Q = \texttt{Sat}(\mathcal{M}, \phi_2) \Leftrightarrow \mathcal{M}, x \models \phi_2 \tag{A.2}$$

For each $x \in X$ we let:

$$\mathcal{I}_x = \{i \in \mathbb{N} | \exists p : x \rightsquigarrow \infty . \mathcal{M}, p(i) \models \neg\phi_1 \wedge \forall j \in \{1, \ldots, i\} . \mathcal{M}, p(j) \models \neg\phi_2\}$$

Note that, by definition, we have that $\mathcal{M}, x \models \phi_1 \mathcal{S} \phi_2$ if and only if $\mathcal{M}, x \models \phi_1$ and $\mathcal{I}_x = \emptyset$.

First we prove that if $\mathcal{I}_x \neq \emptyset$ and $\mathcal{M}, x \models \phi_1$, then x is removed from V at iteration $i = \min \mathcal{I}_x$. This guarantees that if x does not satisfy $\phi_1 \mathcal{S} \phi_2$, then x is eventually removed from V. The proof of this result proceeds by induction on i:

Base of Induction: Let $x \in X$ such that $\mathcal{M}, x \models \phi_1$, $\mathcal{I}_x \neq \emptyset$ and $\min \mathcal{I}_x = 1$. Since $\min \mathcal{I}_x = 1$, we have that there exists $p : x \rightsquigarrow \infty$ such that $\mathcal{M}, p(1) \models \neg\phi_1$ and $\mathcal{M}, p(1) \models \neg\phi_2$. By definition of paths, we also have that $x = p(0)$ and $(x, p(1)) \in R$. This implies that $p(1) \in \mathcal{B}^+(V \cup Q)$ and $x \in pre(p(1))$. By definition of function `CheckSurr` we have that $p(1)$ is in T and x is removed from V during the first iteration. Note that x will be added to T only if it does not satisfy ϕ_2 (i.e. if $x \notin Q$).

Inductive Hypothesis: For each $x \in X$ be such that $\mathcal{M}, x \models \phi_1$, $\mathcal{I}_x \neq \emptyset$ and $\min \mathcal{I}_x = k$, x is removed from V at iteration k.

Inductive Step: Let $x \in X$ be such that $\mathcal{M}, x \models \phi_1$, $\mathcal{I}_x \neq \emptyset$ and $\min \mathcal{I}_x = k + 1$. If $\min \mathcal{I}_x = k + 1$ then there exists $p : x \rightsquigarrow \infty$ such that $\mathcal{M}, p(k + 1) \models \neg\phi_1$ and for each $j \in \{1, \ldots, k + 1\}$ $\mathcal{M}, p(j) \models \neg\phi_2$. We have also that $\mathcal{M}, p(1) \models \phi_1$ (otherwise $\min \mathcal{I}_x = 1$) and $\min \mathcal{I}_{p(1)} = k$ (otherwise $\min \mathcal{I}_x \neq k + 1$). By inductive hypothesis we have that $p(1)$ is removed from V at iteration k. However, since $\mathcal{M}, p(1) \models \neg\phi_2$ we have that $p(1) \notin Q$ and $p(1)$ is in the set T at the beginning of iteration $k + 1$. This implies that $x = p(0)$ is removed from V at iteration $k + 1$, since $x \in pre(p(1))$.

We now prove that if x is removed from V at iteration i, then $\mathcal{I}_x \neq \emptyset$ and $i = \min \mathcal{I}_x$. This ensures that if x is removed from V then x does not satisfy $\phi_1 \mathcal{S} \phi_2$. We proceed by induction on the number of iterations i:

Base of Induction: If $x \in V$ is removed in the first iteration we have that there exists a point $y \in \mathcal{B}^+(V \cup Q)$ such that $(x, y) \in R$. From Equation (A.1) and Equation (A.2) we have that $\mathcal{M}, x \models \phi_1$ while $\mathcal{M}, y \models \neg\phi_1 \wedge \neg\phi_2$. This implies that there exists a path $p : x \rightsquigarrow \infty$ such that $p(1) = y$ and $1 = \min \mathcal{I}_x$.

Inductive Hypothesis: For each point $x \in V$, if x is removed from V at iteration $i \leq k$, then $\mathcal{I}_x \neq \emptyset$ and $i = \min \mathcal{I}_x$.

Inductive Step: Let $x \in V$ be removed at iteration $k+1$. This implies that after k iterations, there exists a point y in T such that $(x, y) \in R$. This implies that y has been removed from V at iteration k and, by inductive hypothesis, $\mathcal{I}_y \neq \emptyset$ and $k = \min \mathcal{I}_y$. Hence, there exists a path $p : y \rightsquigarrow \infty$ such that $\mathcal{M}, p(k) \models \neg\phi_1$ and for each $j \in \{1, \ldots, k\}$ $\mathcal{M}, p(j) \models \neg\phi_2$. Moreover, since $y \in T$, we have also that $y \notin Q$ and, from Equation (A.2), $\mathcal{M}, y \models \neg\phi_2$. We can consider the path $p' : x \rightsquigarrow \infty$ such that, for each j, $p'(0) = x$ and $p'(j+1) = p(j)$. We have that $\mathcal{M}, p'(k+1) \models \neg\phi_1$ and for each $j \in \{1, \ldots, k+1\}$, $\mathcal{M}, p'(j) \models \neg\phi_2$. Hence $\mathcal{I}_x \neq \emptyset$ and $k+1 = \min \mathcal{I}_x$ (otherwise x should be removed from V in a previous iteration).

$\phi = \phi_1 \mathcal{P} \phi_2$**:** We let R_k, T_k and Q_k denote the values of variables R, T and Q at iteration k in `CheckProp`, respectively. Our proof proceeds in three steps. First (**Step 1**) we prove that:

$$\forall k. x \in R_k \wedge post(x) \cap Q_k \neq \emptyset \Rightarrow x \in T_k$$

then (**Step 2**) we show that:

$$\forall k. T_{k+1} = \mathcal{B}^+(R_k) \cap Q_k$$

finally (**Step 3**) we prove that

$$\begin{aligned} \forall k. x \in R_k \quad \Leftrightarrow \quad \mathcal{M}, x \models \Phi_2 \wedge \exists y. \exists l \leq k+1. \exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty. \mathcal{M}, y \models \phi_1 \\ \wedge \forall i. 0 < i < l \implies \mathcal{M}, p(i) \models \phi_2 \end{aligned}$$

After that the statement directly follows from Def. 3.3. However, before proceeding further, we can notice that, by Inductive Hypothesis, the following hold:

$$x \in V = \texttt{Sat}(\mathcal{M}, \phi_1) \Leftrightarrow \mathcal{M}, x \models \phi_1 \tag{A.3}$$

$$x \in Q = \texttt{Sat}(\mathcal{M}, \phi_2) \Leftrightarrow \mathcal{M}, x \models \phi_2 \tag{A.4}$$

Moreover, we can also notice that:

$$\forall k. R_k \cap Q_k = \emptyset \tag{A.5}$$

$$\forall k. R_k \cap Q_k = Sat(\phi_2) \tag{A.6}$$

both the fact above can be derived directly from the definition of `CheckProp` in Fig. 3. Indeed, at the beginning $R_0 = T_0$ while $Q_0 = Sat(\phi_2) \setminus T_0$. Moreover, at every iteration $R_{k+1} = R_k \cup T'$ while $Q_{k+1} = Q_k \setminus T'$.

Step 1: We prove by induction on k that:

$$\forall k.x \in R_k \wedge post(x) \cap Q_k \neq \emptyset \Rightarrow x \in T_k$$

Base of Induction: Let $k = 0$. The statement follows directly from the fact that $R_0 = T_0$. Hence:

$$x \in R_0 \wedge post(x) \cap Q_0 \neq \emptyset \Rightarrow x \in T_0$$

Inductive Hypothesis: For any $k \leq n$:

$$x \in R_k \wedge post(x) \cap Q_k \neq \emptyset \Rightarrow x \in T_k$$

Inductive Step: Let $k = n + 1$:

$x \in R_{n+1} \wedge post(x) \cap Q_{n+1} = \emptyset \wedge x \notin T_{n+1}$

$\iff$ $[\, x \notin T_{n+1} \wedge R_{n+1} = R_n \cup T_{n+1} \,]$

$x \in R_n \wedge post(x) \cap Q_{n+1} = \emptyset$

$\iff$ $[\, Q_{n+1} = Q_n \setminus T_{n+1} \,]$

$x \in R_n \wedge post(x) \cap Q_n = \emptyset$

$\iff$ [Inductive Hypothesis]

$x \in T_n$

$\iff$ [Def. of `CheckProp` in Fig. 3]

$post(x) \cap Q_n \subseteq T_{n+1}$

$\iff$ $[\, Q_{n+1} = Q_n \setminus T_{n+1} \,]$

$post(x) \cap Q_{n+1} = \emptyset \qquad (RAA)$

Step 2: We prove that:

$$\forall k.T_{k+1} = \mathcal{B}^+(R_k) \cap Q_k$$

We first show that $T_{k+1} \subseteq \mathcal{B}^+(R_k) \cap Q_k$. Let $x \in T_{k+1}$

$\iff$ [Def. of `CheckProp` in Fig. 3]

$\exists y \in T_k \wedge x \in post(y) \cap Q_k$

$\implies$ [$T_k \subseteq R_k$ and Def. of $\mathcal{C}$]

$x \in \mathcal{C}(R_k) \wedge x \in Q_k$

$\implies$ [$Eq.\ A.5$]

$x \in \mathcal{C}(R_k) \wedge x \notin R_k \wedge x \in Q_k$

$\implies$ [$Def. of \mathcal{B}^+$]

$x \in \mathcal{B}^+(R_k) \wedge x \in Q_k$

$\implies$ $x \in \mathcal{B}^+(R_k) \cap Q_k$

Now we show that $\mathcal{B}^+(R_k) \cap Q_k \subseteq T_{k+1}$. Let $x \in \mathcal{B}^+(R_k) \cap Q_k$

$\implies$ [Def. of $\mathcal{B}^+$ and Def. of $\mathcal{C}$]

$\exists y \in R_k : x \in post(y) \wedge x \notin R_k \wedge x \in Q_k$

$\implies$ [**Step 1**]

$\exists y \in T_k : x \in post(y) \wedge x \in Q_k$

$\implies$ [Def. of `CheckProp` in Fig. 3]

$x \in T_{k+1}$

Step 3: We can now prove by induction on k that:

$$\forall k.x \in R_k \quad \Leftrightarrow \quad \mathcal{M}, x \models \phi_2 \wedge \exists y.\exists l \leq k+1.\exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty.\mathcal{M}, y \models \phi_1 \wedge \forall i.0 < i < l \implies \mathcal{M}, p(i) \models \phi_2$$

Base of Induction: Let $k = 0$ and $x \in R_0$

$\iff$ [Definition of `CheckProp`]

$x \in \mathcal{C}_R(V) \cap Sat(\phi_2)$

$\iff$ [Definition of $\mathcal{C}_R$]

$x \in (V \cap \mathcal{B}^+(V)) \cap Sat(\phi_2)$

$\iff$ $x \in (Q \cap V) \cup (Q \cap \mathcal{B}^+(V))$

$\iff$ [Definition of $\mathcal{B}^+(V)$]

$x \in (Sat(\phi_2) \cap V)$

or $x \in Sat(\phi_2) \wedge \exists y \in V : (y, x) \in R$

$\iff$ [From A.3 and A.4]

$\mathcal{M}, x \models \phi_2 \wedge \mathcal{M}, x \models \phi_2$

or $\mathcal{M}, x \models \phi_2 \wedge \exists y.\mathcal{M}, y \models \phi_1 : (y, x) \in R$

$\iff$ [From Def. 2.31 and Lemma 2.33]

$\mathcal{M}, x \models \phi_2 \wedge$

$\exists y.\exists l \leq 1.\exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty.\mathcal{M}, y \models \phi_1 \wedge \forall i.0 < i < l+1 \implies \mathcal{M}, p(i) \models \phi_2$

Inductive Hypothesis: For any $k \leq n$:

$$x \in R_k \quad \Leftrightarrow \quad \mathcal{M}, x \models \phi_2 \wedge \exists y.\exists l \leq k+1.\exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty.\mathcal{M}, y \models \phi_1 \wedge \forall i.0 < i < l \implies \mathcal{M}, p(i) \models \phi_2$$

Inductive Step: Let $k = n+1$ and $x \in R_{n+1}$

$\iff$ [Def. of `CheckProp` in Fig. 3]

$x \in R_n \cup T_{n+1}$

$\iff$ $x \in R_n$

or $x \in T_{n+1}$

$\iff$ [**Step 2**]

$x \in R_n$

or $x \in \mathcal{B}^+(R_n) \cap Q_n$

$\iff$ [Def. of $\mathcal{B}^+$]

$x \in R_n$

or $\exists x' \in R_n . x \in post(x') \cap Q_n$

$\iff$ [A.6 and A.4]

$x \in R_n$

or $\exists x' \in R_n . x \in post(x') \wedge \mathcal{M}, x \models \phi_2$

$\iff$ [Inductive Hypothesis]

$\mathcal{M}, x \models \phi_2 \wedge \exists y . \exists l \leq n+1 . \exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty . \mathcal{M}, y \models \phi_1$
$\wedge \forall i . 0 < i < l \implies \mathcal{M}, p(i) \models \phi_2$

or $\exists x' . \in R_n . \mathcal{M}, x' \models \phi_2 \wedge \exists y . \exists l \leq n+1 . \exists p : y \overset{l}{\underset{x'}{\rightsquigarrow}} \infty . \mathcal{M}, y \models \phi_1$
$\wedge \forall i . 0 < i < l \implies \mathcal{M}, p(i) \models \phi_2$

$x \in post(x') \wedge \mathcal{M}, x \models \phi_2$

$\iff$ $\mathcal{M}, x \models \phi_2 \wedge \exists y . \exists l \leq n+1 . \exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty . \mathcal{M}, y \models \phi_1$
$\wedge \forall i . 0 < i < l \implies \mathcal{M}, p(i) \models \phi_2$

or $\mathcal{M}, x \models \phi_2 \wedge \exists y . \exists 0 < l \leq n+2 . \exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty . \mathcal{M}, y \models \phi_1$
$\wedge \forall i . 0 < i < l \implies \mathcal{M}, p(i) \models \phi_2$

$\iff$ $\mathcal{M}, x \models \phi_2 \wedge \exists y . \exists l \leq n+2 . \exists p : y \overset{l}{\underset{x}{\rightsquigarrow}} \infty . \mathcal{M}, y \models \phi_1$
$\wedge \forall i . 0 < i < l \implies \mathcal{M}, p(i) \models \phi_2$ □

Proof. (of Theorem 6.4) We provide a sketch, as the core of the proof is that of Tarjan's algorithm, which we assume given. The proof is by induction on the structure of formulas. The only case where the algorithm is not a direct implementation of its mathematical definition is the one for $\psi = \mathcal{G}\phi$. If $A = \emptyset$ the algorithm returns $True$. This is correct by definition of $\models$, as the empty set is strongly connected. Otherwise, the set of points B satisfying ϕ is computed using function `Sat`, and the algorithm returns *False* if $A \not\subseteq B$. This is correct since all elements of A must satisfy ϕ. Under the hypothesis that $0 \neq A \subseteq B$, an element x is chosen from A, and the algorithm executes a depth-first search according to

[Tar72], modified to only follow successors of x that are in B. Note that the start node x is in B, therefore the algorithm only visits nodes in B. For each strongly connected component C reachable from x in the subgraph defined by B, the algorithm checks whether $A \subseteq C$. If this is the case, then $\mathcal{M}, A \models \mathcal{G}\phi$ and the algorithm returns *True*. Conversely, if there is at least one point in $A \cap C$, but not all points of A are in C, then $\mathcal{M}, A \not\models \mathcal{G}\phi$. To see this, consider $y \in A \cap C$ and $z \in A \cap (X \setminus C)$. It cannot be the case that there are a path from y to z and a path from z to y both only crossing nodes in B, otherwise we would have $z \in C$. Therefore, the algorithm returns *False*. If a strongly connected component is found, but no node of A belongs to it, the algorithm returns *undefined* and the depth-first search continues. One of the first two conditions necessarily happens along the execution of Algorithm 5, when invoked from Algorithm 4, since there is at least one strongly connected component reachable from x and containing x itself, with $x \in A$. Therefore, Algorithm 5 never returns *undefined* when $x \in A$. Termination, and the fact that the algorithm effectively finds strongly connected components, is a consequence of correctness of Tarjan's procedure. The worst case time complexity of Tarjan's algorithm is $\mathcal{O}(|X| + |R|)$ steps. This, the fact that the definition of Algorithm 4 is by induction on the structure of formulas, and Theorem 6.3, cause the algorithm to have time complexity $\mathcal{O}(k \cdot (|X| + |R|))$. □