UvA-DARE (Digital Academic Repository)

Geometric model checking of continuous space

Bezhanishvili, N.; Ciancia, V.; Gabelaia, D.; Grilletti, G.; Latella, D.; Massink, M.

Link to publication

GEOMETRIC MODEL CHECKING OF CONTINUOUS SPACE

NICK BEZHANISHVILI [a], VINCENZO CIANCIA [b], DAVID GABELAIA [c],
GIANLUCA GRILLETTI [d], DIEGO LATELLA [b], AND MIEKE MASSINK [b]

[a] Institute for Logic, Language and Computation, University of Amsterdam, The Netherlands
   e-mail address: n.bezhanishvili@uva.nl

[b] Istituto di Scienza e Tecnologie dell'Informazione "A. Faedo", Consiglio Nazionale delle Ricerche,
   Pisa, Italy
   e-mail address: {vincenzo.ciancia,diego.latella,mieke.massink}@cnr.it

[c] TSU Razmadze Mathematical Institute, Tbilisi, Georgia
   e-mail address: gabelaia@gmail.com

[d] Munich Center for Mathematical Philosophy, Ludwig-Maximilians-Universität München, Germany
   e-mail address: grilletti.gianluca@gmail.com

ABSTRACT. Topological Spatial Model Checking is a recent paradigm where model checking techniques are developed for the topological interpretation of Modal Logic. The Spatial Logic of Closure Spaces, SLCS, extends Modal Logic with reachability connectives that, in turn, can be used for expressing interesting spatial properties, such as "being near to" or "being surrounded by". SLCS constitutes the kernel of a solid logical framework for reasoning about *discrete* space, such as graphs and digital images, interpreted as quasi discrete closure spaces. Following a recently developed *geometric* semantics of Modal Logic, we propose an interpretation of SLCS in *continuous space*, admitting a geometric spatial model checking procedure, by resorting to models based on polyhedra. Such representations of space are increasingly relevant in many domains of application, due to recent developments of 3D scanning and visualisation techniques that exploit mesh processing. We introduce PolyLogicA, a geometric spatial model checker for SLCS formulas on polyhedra and demonstrate feasibility of our approach on two 3D polyhedral models of realistic size. Finally, we introduce a geometric definition of bisimilarity, proving that it characterises logical equivalence.

1. INTRODUCTION AND RELATED WORK

Spatial reasoning and spatial properties are of interest in a vast number of domains, ranging from collective adaptive systems, concerned with the emergence of spatial patterns, and the mobility and distribution of devices in cyber-physical systems to medical imaging and interactive visualisation. Recently, novel variants of model checking have been

developed, moving the focus from checking *temporal* properties to *spatial* properties, see for example [GSC⁺09, CLLM14, CLLM16a, CLLM16b, NBC⁺18, HJK⁺15, MBL⁺21], and, in fact, also to the combination of reasoning on time and space in spatio-temporal model checking [CGL⁺14, CLMP15, CGL⁺15, CGG⁺18, CLM⁺16, Gri16, TKG17].

This so-called *topological* approach to spatial logic and spatial model checking has its origin in the ideas by McKinsey and Tarski [MT44], who recognised the possibility of reasoning about space using topology as a mathematical framework for the interpretation of modal logic (see [BB07] for a thorough introduction). The work by Ciancia et al. (see e.g. [CLLM14, CLLM16a]) builds on these theoretical developments using *Closure Spaces*, a generalisation of topological spaces encompassing also general *discrete* spatial structures such as graphs [Gal99, Gal14], as underlying model for the *Spatial Logic for Closure Spaces* SLCS. The original version of this spatial logic included two spatial operators, the *near* operator and the *surrounded* operator. The points in space satisfying 'near ϕ' are all those points close to any point satisfying ϕ. In other words, the *near* operator is interpreted as a closure operator on space. The points satisfying 'ϕ surrounded by ψ', instead, are all those points satisfying ϕ from which no path can be found that passes by a point not satisfying ϕ without first passing by a point satisfying ψ. In other words, these are those points, satisfying ϕ, that are surrounded by points satisfying ψ.

Two different spatial model-checkers for finite (quasi-discrete) closure spaces were developed based on this foundational work: Topochecker and VoxLogicA.[1,2] These tools have been used successfully in several applications. For example, for the analysis of the spatial and spatio-temporal aspects of excessive delays in returning bikes in the London bike sharing system [CLM⁺16] and the spatio-temporal aspects of the emergence of Turing patterns [CLLM16b, NBC⁺18] and the robustness of their formation in the presence of noise. Also in the field of medical imaging these spatial model checking techniques have shown to be very promising as a novel approach to the segmentation of malignant brain lesions [BCLM19b, BBC⁺20] in 3D MRI scans as well as for the segmentation of normal brain tissue, such as white and grey matter [BCLM19a]. A similar approach has also been used for the segmentation of nevi[3] in 2D dermoscopic images [BBC⁺21]. The application of spatial model checking techniques on publicly available datasets, consisting of hundreds of such medical images, showed that an accuracy and computational efficiency can be obtained that is in line with the state-of-the-art in such fields.

So far, spatial model checking approaches focused on *discrete* spatial structures, i.e., discrete sets of points related by an adjacency relation, namely graphs. However, in several domains, e.g., medical imaging and visualisation in virtual reality, reasoning on *continuous* space is required. For example, computerised 3D visualisation of medical images can help physicians to make better diagnoses or treatment plans. Images used for visualisation often consist of *continuous* spatial structures that are divided into suitable areas of different size using mesh techniques such as *triangular surface meshes* or *tetrahedral volume meshes* (see for example [LPZ12]).

In the present work we therefore focus on the development of the foundations to reason about and model-check properties of *continuous* space. Interestingly, we will also show how our earlier results on discrete space are, in part, also relevant to model-checking

---

[1]Topochecker: a topological model checker, see http://topochecker.isti.cnr.it, https://github.com/vincenzoml/topochecker

[2]VoxLogicA: See https://github.com/vincenzoml/VoxLogicA.

[3]Benign and malignant lesions of the skin.

continuous space. Unlike the topological semantics, where formulas are interpreted in the powerset algebra of a topological space, in the polyhedral semantics formulas are assigned polyhedral subsets of an m-dimensional Euclidean space. Polyhedral subsets can be thought of as finite unions of simplexes (i.e., n-dimensional triangles). Using piecewise linear geometry (triangulations, nerves), the work in [BMMP18] gives a full characterization of the intuitionistic and modal logics of the class of all compact polyhedra and [ABGM21, AD19] provide an infinite family of polyhedrally complete modal and intermediate logics.

The version of SLCS we use in this paper is obtained by extending the polyhedral modal logic with a *spatial reachability* modality γ, which is a variant of the ρ operator originally proposed in [BCLM19b] that, in turn, can be seen as a spatial version of the Existential Until operator of temporal logics (for more examples of spatial reachability operators, see also [NBBL22]). Roughly speaking, $\gamma(\phi, \psi)$ (pronounced as ψ is reachable through ϕ) means that a point satisfying ψ is reachable by a path satisfying ϕ along the way. The reachability modality is quite expressive and other operators, relevant for the intended applications (such as "surrounded", or "grow", discussed in more detail throughout the paper), can be defined based on it. We show that the reachability modality can be defined for polyhedral models.

One of the main conceptual results of the present paper is that, starting from a polyhedral model $\mathcal{X}$, we can build a relational (Kripke) model $\mathcal{M}(\mathcal{X})$ satisfying the same formulas of the logic. In particular, $\mathcal{M}(\mathcal{X})$ is isomorphic to the face poset of an underlying triangulation of $\mathcal{X}$. Triangulation is a standard technique of piecewise linear geometry in which each polyhedron is decomposed in simplexes. That triangulations play an important role in the logical analysis of polyhedra has already been observed in [BMMP18, AD19, ABGM21]. However, here we show this also for the language enriched with the reachability modality γ. In particular, $\mathcal{M}(\mathcal{X})$ captures all the properties expressible in this extended language.

The finite state, Kripke-style semantics that we define preserves all the information that can be discerned by SLCS formulas. This is the key for introducing a novel *geometric model checking* technique to analyse continuous space. We prove that the continuous model of the extended language can be turned into a finite relational model for the same language without losing any of the logical information. A model checking algorithm, along the lines of [BCLM19b], has been implemented in the free and open source geometric model checker PolyLogicA, which brings the core features of VoxLogicA (global model checking, concurrent multi-core execution, "memoization" at the syntactic level) to the continuous space domain.

Finally, we define a notion of *bisimilarity* between two polyhedral models, and we prove that bisimilarity preserves and reflects logical equivalence.

Summarizing, the original contributions in this paper are:

- The development of a (continuous space) polyhedral semantics for the logic SLCS building upon recent developments in polyhedral semantics for modal and intuitionistic logic [BMMP18, ABGM21, AD19];
- The introduction of a *finite* state relational Kripke model $\mathcal{M}(\mathcal{X})$ for any given (continuous space) polyhedral model $\mathcal{X}$, such that $\mathcal{M}(\mathcal{X})$ provides a full logical invariant for $\mathcal{X}$ with respect to SLCS;
- A novel geometric spatial model checking algorithm for the verification of SLCS properties on polyhedral models;
- First proof-of-concept model checking of two realistic-size polyhedral models using Poly-LogicA, the first-in-its-kind prototype implementation of the geometric model checking algorithm;

• A novel notion of *simplicial bisimilarity* characterising logical equivalence for SLCS formulas.

Further related work. The theoretical framework for spatial model checking of continuous space in the present paper is based on spatial models involving polyhedra (see also the work in [CM21], generalising the semantics of SLCS to a categorical setting based on hyperdoctrines). Polyhedra also play an important role in development of model checking algorithms for the verification of behavioural properties of real-time and hybrid systems (see for example [HH94, Alu11, BFGH17, Hen00, AGH+19] and references therein). In that context polyhedra, and their related notions such as template polyhedra [SDI08, BFGH17] and zonotopes [GL08], are obtained from sets of linear inequalities involving real-time constraints on system behaviour and are a natural representation of sets of states of such systems. In the present paper we focus on *spatial* properties of continuous space rather than behavioural properties.

Topology and simplicial complexes also play an important role in the domain of geographic information systems (GIS). In that domain of application simplicial complexes are used as an efficient data structure to store large geospatial data sets [BBJ+20] in 2D or 3D. They also form the core of several important tools in this domain such as the GeoToolKit [BBB+04]. This could therefore be in principle another interesting domain of application for spatial model checking techniques, enriching the spatial query languages that are currently used in this database oriented domain.

Model Checking of simplicial complexes is also the topic of a recent, independently developed result by Loreti and Quadrini [LQ21]. Therein, a logic is defined, sharing similar syntax, but *not its semantics*, with the logic SLCS used in the present paper. More precisely, the domain upon which formulas are interpreted, are not (sets of) points in polyhedra, but rather (sets of) simplexes. This is reflected by the choice of the adjacency relation between objects[4] giving rise to *nearness* and *reachability*. In contrast, the domain of interpretation of our spatial logic SLCS are points of topological spaces, with polyhedral atomic valuations, and reachability is the classical topological notion, defined via paths.

Looking at tools, perhaps closest to our work is the python library `pymeshlab` [MC21], which is able to programmatically modify 3D meshes based on pre-built operators (mostly traditional 3D imaging filters). Since also the selection of connected components is an operator of the library, we envisage that it could be possible, given an SLCS formula, to encode it as a python program containing `pymeshlab` functions, effectively using the library as a backend for VoxLogicA. Indeed, using such a library within the context of a traditional programming language does not lay at the same level of abstraction of using a simple declarative language such as SLCS, with automatic parallelisation, and automatic memoization (caching) of intermediate results, so a direct comparison of the library and our tool would be misleading.

Outline. Section 2 introduces the basic geometrical notions and notation. Section 3 recalls SLCS and provides its semantics on polyhedral models. Section 4 and Section 5 present the foundations for geometric model checking and the related model checker PolyLogicA, respectively, including two proof-of-concept examples on realistic size polyhedra. In Section 6 the concept of simplicial bisimilarity is introduced and it is shown that it characterises logic

---

[4]Actually, the paper proposes three different relations, see Definition 1–3 therein.

equivalence for SLCS formulas. Section 7 concludes the paper with an outlook for future work. Proofs of the most relevant theoretical results are reported in Appendix A.

2. Background

In this section, we establish the basic geometric notions that we use in this work. See [Mau80, Chapter 2] for more details on these matters. In particular, we recall simplexes and simplicial complexes. They are frequently used in, for example, computer graphics and simulation. Polyhedra are the set-theoretic union of simplicial complexes and form the spatial models for the polyhedra model checking approach proposed in subsequent sections.

Definition 2.1 (Simplex). A *d-simplex* σ is the convex hull of a finite set $V = \{v_0, v_1, \ldots, v_d\}$ $\subseteq \mathbb{R}^m$ of $d+1$ affinely independent points,[5] that is the set

$$\sigma = \left\{ \lambda_0 v_0 + \cdots + \lambda_d v_d \;\middle|\; \forall i . \lambda_i \in [0,1] \text{ and } \sum_{i=0}^d \lambda_i = 1 \right\}.$$

The number d is called the *dimension* of σ and $v_0, \ldots, v_d$ are called its *vertices*.

In Definition 2.1, any subset of $\{v_0, \ldots, v_d\}$ is also a set of affinely independent points, and thus it spans a simplex τ: we call τ a *face* of σ (in symbols $\tau \preceq \sigma$), and we call it a *proper face* if $\tau \neq \emptyset$ and $\tau \neq \sigma$.

Simplexes are, informally speaking, the simplest linear convex bounded shapes. A two-dimensional simplex is a triangle; a three-dimensional simplex is a tetrahedron, and so on. Note that the two-dimensional faces of a tetrahedron are triangles, whose sides are line segments (one-dimensional simplexes), whose faces (endpoints) are points (zero-dimensional simplexes).

Next, we identify the "internal part" of a simplex.

Definition 2.2 (Relative interior). In terms of the notation of Definition 2.1, let the *relative interior* of σ be the set $\widetilde{\sigma} := \left\{ \sum_{i=0}^d \lambda_i v_i \;\middle|\; \forall i . \lambda_i \in (0,1] \text{ and } \sum_{i=0}^d \lambda_i = 1 \right\}$.

Note that if σ is non-empty then also $\widetilde{\sigma}$ is non-empty. For instance, $b_\sigma := \sum_{i=0}^d \frac{1}{d+1} v_i$ (i.e., the *barycentre* of σ) is an element of $\widetilde{\sigma}$. In particular, the relative interior of a point p is p itself, and the relative interior of the empty simplex is the empty simplex itself.

We emphasize another interesting property, which follows from the definition of face and that of relative interior: Each simplex σ is partitioned by the relative interiors of its faces, that is, $\sigma = \bigcup \{\widetilde{\tau} \mid \tau \preceq \sigma\}$. For example, a triangle can be partitioned into its interior (an open triangle), three open segments (sides without endpoints) and the three vertices.

In this paper we adopt the Kuratowski axiomatic characterisation of topological spaces, based on the closure operator. We denote the latter by $\mathcal{C}$. Given topological space $(X, \mathcal{C})$ we let $\mathcal{I}$ denote the interior operator, i.e., the dual of $\mathcal{C}$ defined as $\mathcal{I}(A) = X \setminus \mathcal{C}(X \setminus A)$ for all $A \subseteq X$. Finally, for $X' \subseteq X$, the topological sub-space of $(X, \mathcal{C})$ generated by X' is the pair $(X', \mathcal{C}')$, where $\mathcal{C}'(A) = \mathcal{C}(A) \cap X'$ for all $A \subseteq X'$. For ease of notation, we will indicate the topological space $(X, \mathcal{C})$ simply as X when no confusion arises.

Simplexes are bounded, convex, compact subspaces of $\mathbb{R}^m$ [Mau80, Proposition 2.3.3]. Being subsets of an Euclidean space $\mathbb{R}^m$, a simplex σ inherits the topological structure of

---

[5]$v_0, \ldots, v_d$ are *affinely independent* if $v_1 - v_0, \ldots, v_d - v_0$ are linearly independent. In particular, this condition implies that $d \leq m$.

Figure 1: On the left, a graphical representation of a simplicial complex is shown in grey and black. On the right, we highlight some of the faces of the simplicial complex, of dimension 2, 1 and 0 (the green triangle, blue segment and red point, respectively).
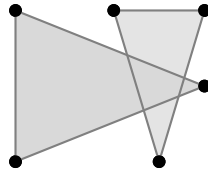

Figure 2: A graphical representation of a collection of simplexes (in 2D) *not* forming a simplicial complex. In particular, the collection of simplexes depicted consists of 6 points, 6 edges and 2 triangles. Notice that Condition (2) of Definition 2.3 is not respected: the intersection between the two triangles is not a simplex of the collection itself.

the sub-space from $\mathbb{R}^m$. Let us indicate by $\mathcal{C}_m$ the closure operator of $\mathbb{R}^m$ and by $\mathcal{C}_\sigma$ the induced closure operator on σ. Notice that, since σ is a closed subset of $\mathbb{R}^m$, the closure of any subset $A \subseteq \sigma$ is the same computed in both topologies, that is, $\mathcal{C}_m(A) = \mathcal{C}_\sigma(A)$. An emblematic example is the closure of $\widetilde{\sigma}$: $\mathcal{C}_\sigma(\widetilde{\sigma}) = \mathcal{C}_m(\widetilde{\sigma}) = \sigma$. However, the interior of a set may depend on the topology considered: for example, $\mathcal{I}_m(\sigma)$ is either $\widetilde{\sigma}$ if the dimension of σ is m, or the empty set otherwise. On the other hand, $\mathcal{I}_\sigma(\sigma) = \sigma$, independently from the dimension.

More complex spaces are obtained by "gluing" simplexes together.

Definition 2.3 (Simplicial complex). A *simplicial complex* K is a finite set of simplexes of $\mathbb{R}^m$ such that:

(1) If $\sigma \in$ K and τ is a face of σ, then $\tau \in$ K;
(2) If $\sigma, \tau \in$ K, then $\sigma \cap \tau$ is a face of σ and τ (possibly the empty simplex).

Figure 1 illustrates an example of a simplicial complex K and highlights some of the simplexes comprising it. Figure 2 illustrates a collection of simplexes that *does not* form a simplicial complex. The *dimension* of K is the maximum of the dimensions of its simplexes, i.e., 2 in Figure 1. The face relation on simplexes is inherited by simplicial complexes in the expected way: the face relation $\preceq$ on simplicial complex K is the union of the face relations on the simplexes composing K.

Given simplicial complex K $= \{\sigma_1, \ldots, \sigma_n\}$, the *polyhedron* of K, denoted by $|$K$|$, is the set-theoretic union of the simplexes in K. Formally, $|$K$| := \bigcup_{i=1}^n \sigma_i$. Figure 3a shows a simplicial complex, which is a set of 11 elements – 4 points, 5 open segments and 2 open triangles. The set-theoretic union of these 11 simplexes is the square shown in Figure 3c – the corresponding polyhedron. Note that a different simplicial complex, shown in Figure 3b, has the same (underlying) polyhedron.
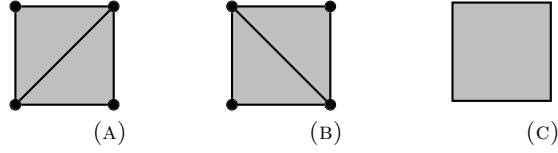
Figure 3: Two different simplicial complexes, (3a) and (3b) leading to the same polyhedron (3c)

As in the case of simplexes, the polyhedron |K| is a subset of the ambient space $\mathbb{R}^m$ and so inherits the topological structure of sub-space of $\mathbb{R}^m$. Let us indicate with $\mathcal{C}_K$ the corresponding closure operator — notice that this operator only depends on the set |K|. A point of |K| may belong to several of the simplexes in K. However, there is a natural way to associate to each point of |K| the "smallest" simplex it belongs to.

Lemma 2.4. *Each point of* |K| *belongs to the relative interior of exactly one non-empty simplex in* K. *That is,* $\widetilde{K} := \{\widetilde{\sigma} \,|\, \sigma \in K \setminus \{\emptyset\}\}$ *is a partition of* |K|.

We call $\widetilde{K}$ a *simplicial partition* of |K|, and we call its elements the *cells* of the partition.[6] Note that distinct simplicial complexes induce distinct partitions, even when they are associated to the same polyhedron, as shown in Figure 3. From now on, to ease readability, we fix a simplicial complex K, with the associated |K| and $\widetilde{K}$.
Finally, we recall the topological notion of *path*.

Definition 2.5. A topological *path* in a topological space P is a total, continuous function $\pi : [0,1] \to P$, where $[0,1]$ is equipped with the subspace topology of $\mathbb{R}$.

With a mild abuse of notation, for S a subset of $[0,1]$ and π a path, we write $\pi(S)$ to denote $\{\pi(x) \mid x \in S\}$.

3. Interpreting SLCS on Polyhedra

In this section we introduce the main theory driving our model checking approach to polyhedra. In the classical topological tradition, valuations of atomic propositions can be arbitrary subsets of the space. In this work, instead, we restrict our attention to a specific class of spatial regions, namely unions of cells of a fixed simplicial partition. This simple change makes it possible to define the spatial logic SLCS on continuous space, while retaining decidability of the model checking problem. First of all, we introduce the syntax of the variant of SLCS that we use in this paper, that is based on the binary modality γ instead of ρ of [BCLM19b]; the relationship between ρ and γ will be shown in Proposition A.1.

Definition 3.1 (Syntax)**.** The syntax of the logic SLCS is:

$$\phi \ ::= \ \top \mid p \mid \neg\phi \mid \phi \wedge \phi \mid \Box\phi \mid \gamma(\phi, \phi)$$

where p is an atomic proposition, taken from a fixed finite set AP.

---

[6]We use the terminology *cells* in this way for the purposes of this paper; there is no relation between such cells and the so-called *cell complexes* of algebraic topology.

Thus, we enhance the basic modal logic with a *spatial reachability* operator γ. As in the standard topological semantics for modal logic, we interpret formulas as sets of points. Boolean operators are given their standard set-theoretical interpretation; disjunction $\vee$ is derived via the De Morgan laws. The $\Box$ modality corresponds to *topological* interior $\mathcal{I}$.

The formula $\gamma(\phi, \psi)$ ("reach ψ through ϕ") is satisfied by a point if there is a path rooted in that point, leading to a point satisfying ψ and whose intermediate points all satisfy ϕ. We provide a number of examples of the operators after presenting the models and formal semantics of the logic. In the following, we let $\mathcal{P}(P)$ denote the powerset of P.

Definition 3.2 (Model)**.** A *Polyhedral Model* is a triplet $\mathcal{X} = \langle P, \mathrm{K}, V \rangle$, where $P \subseteq \mathbb{R}^d$ is a polyhedron, K is a simplicial complex such that $P = |\mathrm{K}|$, and $V : \mathrm{AP} \to \mathcal{P}(P)$ is a valuation such that $V(p)$ is a union of cells of $\widetilde{\mathrm{K}}$.

Polyhedral models are essentially topological models with some extra restrictions on the valuation: P plays the role of the topological space and V is used to interpret atomic propositions as specific subsets of this space, namely those that are the union of a finite number of simplicial cells. From now on, fix a polyhedral model $\mathcal{X} = \langle P, \mathrm{K}, V \rangle$.

Definition 3.3 (Semantics)**.** Given $x \in P$, satisfaction $\mathcal{X}, x \vDash \phi$ over formulas ϕ is given by the following inductive clauses, where we let $\llbracket \phi \rrbracket^{\mathcal{X}}$ denote the set $\{ x \in P \mid \mathcal{X}, x \vDash \phi \}$:

$$\begin{aligned}
&\mathcal{X}, x \vDash \top && \text{always holds} \\
&\mathcal{X}, x \vDash p && \iff && x \in V(p) \text{ for } p \in \mathrm{AP} \\
&\mathcal{X}, x \vDash \neg\phi && \iff && \mathcal{X}, x \nvDash \phi \\
&\mathcal{X}, x \vDash \phi \wedge \psi && \iff && \mathcal{X}, x \vDash \phi \text{ and } \mathcal{X}, x \vDash \psi \\
&\mathcal{X}, x \vDash \Box\phi && \iff && x \in \mathcal{I}_P(\llbracket \phi \rrbracket^{\mathcal{X}}) \\
&\mathcal{X}, x \vDash \gamma(\phi, \psi) && \iff && \text{there exists a path } \pi \text{ such that} \\
& && && \pi(0) = x, \ \pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}} \text{ and } \pi((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}
\end{aligned}$$

The definition of the satisfaction relation for the standard operators of modal logic is the usual one for the classical topological interpretation. In particular, note the interpretation of $\Box\phi$ as the *topological interior* with respect to the topology of $P = |\mathrm{K}|$, intuitively expressing that point x is in the "internal" part of the set of points satisfying ϕ. Notice that the closure operator $\mathcal{C}_P$ can be obtained as the dual of topological interior $\Diamond\phi = \neg(\Box\neg\phi)$. Figure 4 illustrates these operators and their combination applied on a simple polyhedral model.

Regarding spatial reachability, a point x satisfies $\gamma(\phi, \psi)$ in model $\mathcal{X}$ if there is a path π rooted in x leading to a point y satisfying ψ; in addition, all the points that lay in π, except x and y, are required to satisfy ϕ. Indeed, several different variants of reachability could be defined using this operator.

As a prominent example, the reachability modality $\rho\,\psi\,[\phi]$ introduced in [BCLM19b], that we also employ to introduce some derived operators, can be defined as $\rho\,\psi[\phi] := \psi \vee \gamma(\phi, \psi)$. Actually, the two operators are inter-definable, by letting $\gamma(\phi, \psi) := \rho\,(\phi \wedge \rho\,\psi[\phi])[\phi]$ (see Proposition A.1). In this work we opt to use γ since, in the context of polyhedral models, its definition is more concise.

Another relevant spatial modality is the *surrounded* operator $\mathcal{S}$ (e.g., [CLLM16a, LPS20, TKG17, NBC$^+$18] use it as a primitive of the language). A point x satisfies $\phi\,\mathcal{S}\,\psi$ if it lays in an area whose points satisfy ϕ, and that is limited (i.e., surrounded) by points that satisfy ψ. In other words, it is not possible to exit this area without passing by a point satisfying ψ. Following [BCLM19b], we can define the operator $\mathcal{S}$ on polyhedral models in terms of ρ through the following expression: $\phi \wedge \neg\rho\,(\neg(\phi \vee \psi))[\neg\psi]$.

(A)
Model A

(B)
$\Box g$

(C)
$\Diamond g = \neg(\Box \neg g)$
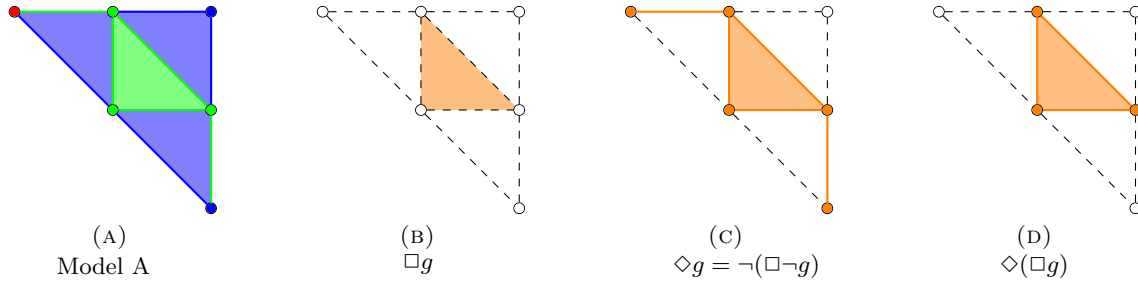
(D)
$\Diamond(\Box g)$

Figure 4: Examples of basic topological operators on Polyhedra. 4a) Polyhedral model A. Circles denote 0-dimensional simplexes. The valuation of atomic propositions r, g, b in model A is given by the colours *red*, *green*, and *blue*, respectively. The points in model A that satisfy the following example formulas are shown in orange in: 4b) $\Box g$, i.e., points satisfying the *topological interior* of the part in *green* in model A; 4c) $\Diamond g$, i.e., points satisfying the *topological closure* of the part in *green* in model A; 4d) $\Diamond(\Box g)$, i.e., points satisfying the closure of the interior of the part in *green* in model A.

(A)
Model C

(B)
$\gamma(g, b)$

(C)
$grow(r, g)$

(D)
$g \wedge \gamma(g, \gamma(b, r))$
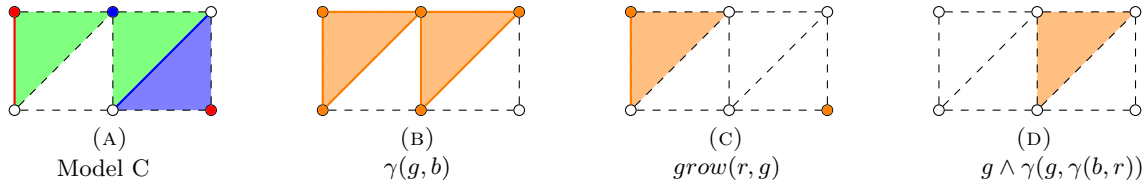
Figure 5: Examples illustrating the reachability operator on Polyhedra. 5a) Polyhedral model C. Circles and colours have the same meaning as in Figure 4. Dashed segments and white points and triangles do not satisfy any atomic proposition. 5b) In orange: Points satisfying $\gamma(g, b)$ applied to the model in 5a. Note that these include points in the closure of the green area since the first point of the paths passing by green and reaching blue do not need to be green themselves. 5c) In orange: Points satisfying $grow(r, g)$. Note that only the points corresponding to the red area and one green triangle in the model in 5a satisfy this formula. 5d) In orange: Points satisfying $g \wedge \gamma(g, \gamma(b, r))$; by nesting reachability, quite complex spatial formulas may be defined.

Some examples involving the reachability operator are shown in Figure 5. We refer to the caption of that figure for more detailed explanation. Notably, we illustrate the derived operator $grow(a, b)$, that also played an important role in the brain tumour segmentation procedure presented in [BCLM19b]. The operator *grow* is reminiscent of the technique of *region growing* in Medical Imaging, and it is used to characterise those areas of space satisfying b that are in contact with areas of space satisfying a, or, in other words, the operator lets a "grow" inside b (and no further). The formal definition is $grow(\phi_1, \phi_2) := \phi_1 \vee (\phi_2 \wedge \rho\, \phi_1[\phi_2])$.

Note that the same polyhedron P can be associated with different simplicial complexes: our semantics is not sensitive to such presentational ambiguity in the description of P.[7] This is because, although we need to specify K to spell out the restriction on the range of V, K itself does not play a role in the semantics, as shown in the following proposition.

Lemma 3.4. *Let $\mathcal{X} = \langle P, \mathrm{K}, V \rangle$ and $\mathcal{X}' = \langle P, \mathrm{K}', V \rangle$ be two models sharing the same P and V. For each $x \in P$ and ϕ we have: $\mathcal{X}, x \vDash \phi \iff \mathcal{X}', x \vDash \phi$.*

Therefore, for the sake of readability, we will sometimes indicate a polyhedral model with the notation $\mathcal{X} = \langle P, V \rangle$, abstracting from the particular choice of K. Nevertheless, we require V to range over unions of cells of *some* polyhedral partition, thus restricting the semantics to spatial regions definable in terms of polyhedra. We will call a simplicial complex K as in Definition 3.2 *coherent* with the model $\mathcal{X} = \langle P, V \rangle$.

We mentioned that employing polyhedra allows for a *finitary treatment* of the semantics. The following results are essential to formalize this intuition, which will be further investigated in Section 4.

Definition 3.5. Let $\mathcal{X} = \langle P, V \rangle$ be a polyhedral model. *Logical equivalence* $\equiv$ is the binary relation on P such that $x \equiv y$ if and only if, for every formula ϕ: $\mathcal{X}, x \vDash \phi \iff \mathcal{X}, y \vDash \phi$.

Lemma 3.6. *Let $\mathcal{X}$ be a polyhedral model and K a simplicial complex coherent with $\mathcal{X}$. Then for each cell $\widetilde{\sigma} \in \widetilde{\mathrm{K}}$ and $x, y \in \widetilde{\sigma}$ we have $x \equiv y$.*

In particular, for every formula ϕ, $[\![\phi]\!]^{\mathcal{X}}$ is a (finite) union of cells of $\widetilde{\mathrm{K}}$.

Proposition 3.7. *Given a polyhedral model $\mathcal{X}$, the relation $\equiv$ has only finitely many equivalence classes. Furthermore, each equivalence class C has a characteristic formula ϕ^C such that $\mathcal{X}, x \vDash \phi^C \iff x \in C$.*

The above facts are also useful to prove an interesting feature of polyhedral models, namely that the $\square$ modality can be considered a derived operator, since it is expressible using γ. This considerably simplifies proofs.

Theorem 3.8. *For each formula ϕ, we have $\mathcal{X}, x \vDash \square\phi \iff \mathcal{X}, x \vDash \neg\gamma(\neg\phi, \top)$.*

Another property of polyhedral models which turns out to be fundamental in this work is that we can restrict our attention to a special class of paths — rather than arbitrary paths — to study the reachability operator γ: *piecewise linear paths*.

Definition 3.9 (PL-path). We call a path $\pi : [0, 1] \to P$ *piecewise linear* (or simply PL-path) if there exist values $r_0 = 0, r_1, \ldots, r_k = 1$ such that for every $i = 0, \ldots, k-1$ and $t \in [0, 1]$: $\pi\big(tr_i + (1-t)r_{i+1}\big) = t\pi(r_i) + (1-t)\pi(r_{i+1})$.

Figure 6 shows two examples of piecewise linear paths. We indicate that a path is piecewise linear with the notation $\pi : [0, 1] \overset{PL}{\to} P$. Intuitively, a PL-path is obtained by connecting a finite number of segments and parametrizing them in a suitable way. Even if PL-paths are much simpler than arbitrary paths, when it comes to connectivity in polyhedral models the two classes are interchangeable, as shown in the following lemma.

Lemma 3.10. *Let K be a simplicial complex and $x, y \in |\mathrm{K}|$. Then there exists a path in $|\mathrm{K}|$ from x to y if and only if there is a PL-path in $|\mathrm{K}|$ from x to y.*

---

[7]Such ambiguity can be thought of as being similar, in spirit, to the infinitely many different programs that may result in equivalent Kripke frames, in a specification language for classical model checking applications.
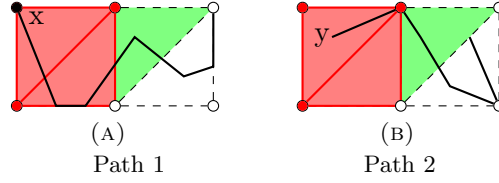
(A)   (B)
Path 1   Path 2

Figure 6: Two examples of piecewise linear paths, one starting in point x and the other in point y.

Using the previous result, we can give an alternative semantic characterization of the reachability operator, which is relevant for the proofs of decidability of model checking (Section 4) and the characterisation of logical equivalence via bisimilarity (Section 6).

Lemma 3.11. *We have:* $\mathcal{X}, x \vDash \gamma(\phi, \psi)$ *if and only if there is a PL-path* $\pi : [0,1] \overset{PL}{\to} P$ *such that* $\pi(0) = x$ *and* $\pi((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ *and* $\pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$.

We conclude this section by pointing out an interesting property of PL-paths in connection to simplicial complexes. By definition, PL-paths are obtained by concatenating a finite number of line segments, but in general each segment might traverse different cells of the simplicial complex K. However we can give an alternative decomposition of these paths so that each portion is fully contained in a unique cell; paths that admit such a decomposition are called *simplicial paths*.

Definition 3.12. A path $\pi : [0,1] \to P$ is *simplicial* if and only if there is a finite sequence $s_0 = 0 < \cdots < s_k = 1$ of values in $[0,1]$ and cells $\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_k \in \widetilde{K}$ such that, for all $i = 1, \ldots, k$, we have $\pi((s_{i-1}, s_i)) \subseteq \widetilde{\sigma}_i$.

Notice that the property of being *simplicial* depends on the simplicial complex K, since a path might be simplicial with respect to a certain triangulation of P, but not with respect to another. For PL-paths, this property holds independently from the simplicial complex considered.

Lemma 3.13. *Any piecewise linear path is simplicial.*

4. Geometric Model Checking

Given a polyhedral model $\mathcal{X}$, this section is devoted to identifying a corresponding Kripke-style, finite model $\mathcal{M}(\mathcal{X})$. Notably, $\mathcal{M}(\mathcal{X})$ is also a topological model in the sense of [BB07] when equipped with the Alexandrov topology, and it is a quotient of $\mathcal{X}$ that preserves and reflects the semantics of each formula. The goal of this section is to extend the standard Kripkean semantics of modal logic to the language of SLCS, by defining a suitable semantics for γ and by showing that $\mathcal{X}$ and $\mathcal{M}(\mathcal{X})$ are logically equivalent, in the sense that, for all $x \in \mathcal{X}$ and for all formulas Φ, letting $\widetilde{\sigma}$ be the only cell such that $x \in \widetilde{\sigma}$, we have: $\mathcal{X}, x \models \Phi$ if and only if $\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \models \Phi$. To do so, we introduce a suitable notion of path in $\mathcal{M}(\mathcal{X})$ corresponding to a simplicial path in $\mathcal{X}$. Model checking on $\mathcal{X}$ can then be carried out using $\mathcal{M}(\mathcal{X})$.

Definition 4.1. Given a polyhedral model $\mathcal{X} = \langle P, K, V \rangle$, we define the Kripke model $\mathcal{M}(\mathcal{X}) = \left\langle \widetilde{K}, \widetilde{\preceq}, \widetilde{V} \right\rangle$, where

(A)
Polyhedral model $\mathcal{X}$

(B)
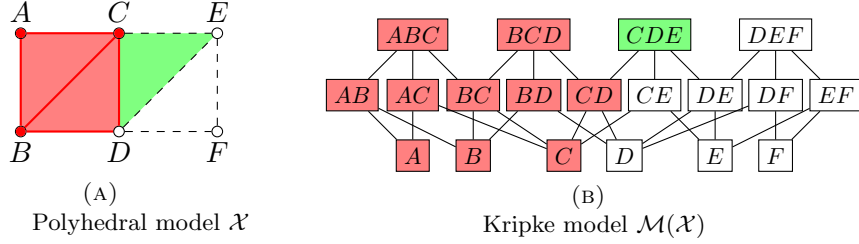Kripke model $\mathcal{M}(\mathcal{X})$

Figure 7: The polyhedral model $\mathcal{X}$ of Figure 17 (7a) and its corresponding Kripke model $\mathcal{M}(\mathcal{X})$ (7b). We indicate a cell by the set of the vertices of the corresponding simplex. The accessibility relation $\widetilde{\preceq}$ is represented via its Hasse diagram (reflexive and transitive edges are omitted). The atomic propositions g and r are indicated in green and red respectively.

- $\widetilde{K}$ is the simplicial partition of $|K|$ generated by K, as Defined in Lemma 2.4,
- $\widetilde{\preceq} \subseteq \widetilde{K} \times \widetilde{K}$ with $\widetilde{\sigma}_1 \widetilde{\preceq} \widetilde{\sigma}_2$ iff $\sigma_1 \preceq \sigma_2$, and
- $\widetilde{\sigma} \in \widetilde{V}(p)$ iff $\widetilde{\sigma} \subseteq V(p)$

where $\preceq$ is the face relation of the simplicial complex K.

Notice that, since $\preceq$ is reflexive, anti-symmetric and transitive, then so is $\widetilde{\preceq}$. An example of a 2D polyhedral model together with its corresponding Kripke model $\mathcal{M}(\mathcal{X})$ is depicted in Figure 7 (where reflexive and transitive relations are omitted). Each cell is identified by the vertices of the corresponding simplexes indicated in the polyhedron on the left in Figure 7.

Such Kripke models can be depicted as a number of rows. The nodes in the bottom row represent the zero-dimensional cells (i.e., the vertices), the middle row the one-dimensional ones (i.e., the line segments) and the top-row the two-dimensional cells (i.e., the triangles). To represent 3D polyhedral models we would have one more row representing the three-dimensional cells (i.e., tetrahedrons), and so on for higher dimensions. We emphasize that $\mathcal{M}(\mathcal{X})$, although disregarding much of the information about a polyhedral model (e.g., the position and size of the simplexes) encodes all the information which is expressible using SLCS, while being a finite — thus computationally tractable — representation of $\mathcal{X}$.

By definition of the relation $\widetilde{\preceq}$, we have that for two different cells $\widetilde{\sigma}_1$ and $\widetilde{\sigma}_2$, $\widetilde{\sigma}_1 \widetilde{\preceq} \widetilde{\sigma}_2$ entails that the two cells are spatially adjacent — more precisely, σ_1 is part of the *boundary* of σ_2. This becomes particularly relevant when studying PL-paths in P. In fact, a PL-path can transit between two different cells $\widetilde{\sigma}_1$ and $\widetilde{\sigma}_2$ only if either $\widetilde{\sigma}_1\widetilde{\preceq}\widetilde{\sigma}_2$ or $\widetilde{\sigma}_1\widetilde{\succeq}\widetilde{\sigma}_2$ holds. Based on this intuition, the next definition introduces special paths on these Kripke models corresponding to PL-paths on the polyhedron.

Definition 4.2 ($\pm$-path). Given a polyhedral model $\mathcal{X}$, with $\mathcal{M}(\mathcal{X}) = \left\langle \widetilde{K}, \widetilde{\preceq}, \widetilde{V} \right\rangle$ as in Definition 4.1, let $\preceq^{\pm}$ be the relation $\widetilde{\preceq} \cup \widetilde{\succeq}$. We say that $\pi : \{0, \ldots, k\} \to \widetilde{K}$ is a $\pm$-*path* (and we indicate it with $\pi : \{0, \ldots, k\} \xrightarrow{\pm} \widetilde{K}$) if $k \geq 2$ and $\pi(0) \widetilde{\preceq} \pi(1) \preceq^{\pm} \pi(2) \preceq^{\pm} \ldots \preceq^{\pm} \pi(k-1) \widetilde{\succeq} \pi(k)$.

Intuitively, a $\pm$-path π represents the cells traversed by a PL-path π' in P; $\pi(0)$ is the cell containing the first point $\pi'(0)$ of the PL-path π'; $\pi(1)$ is the cell containing the the next portion of PL-path π', i.e., the image via the PL-path π' of an interval of the
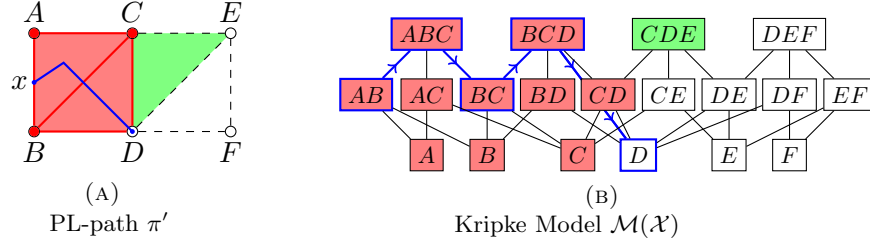
Figure 8: On the left, a PL-path π' (in blue); on the right, the corresponding $\pm$-path π (again in blue). Only $\pi'(0)$ belongs to the simplex AB and the path enters immediately into the cell $\widetilde{ABC}$: this is possible since $\widetilde{AB} \widetilde{\preceq} \widetilde{ABC}$. Likewise, the path ends with a transition from the cell $\widetilde{BCD}$ to $\widetilde{D}$, which is possible since $\widetilde{BCD} \widetilde{\succeq} \widetilde{D}$. These are exactly the requirements on the first and last steps of a $\pm$-path.

form $(0, s)$ or $(0, s]$, for some s, and so on, up to cell $\pi(k)$ that contains the end-point $\pi'(1)$ of the PL-path π'. An example of a PL-path and its corresponding $\pm$-path is shown in Figure 8. Notice that the first step $\pi(0)\pi(1)$ in the $\pm$-path π reflects the move from the starting point in the PL-path to the subsequent segment on the PL-path in polyhedron P. The starting point must obviously be 'connected' to this subsequent path segment. This is only possible if the starting point is in the closure of the cell that contains the interval, in other words, $\pi(0) \widetilde{\preceq} \pi(1)$, that is $\pi(0)$ must be on the boundary of $\mathcal{C}_P(\pi(1))$. Moreover, if $\pi(0) = \pi(1)$, then, depending on the dimension of the cell containing the starting point, the subsequent portion of the path can be in the *same* cell as the starting point. This is for example the case in path y in Figure 6. Obviously, the path segment following the starting point cannot be part of a cell that, in the poset, is strictly below the cell containing the starting point, because then the point and the segment cannot be connected to each other. A similar situation holds for the end-point of the PL-path and the last segment of the path leading to it. In that case the end-point should be in a cell that is in the closure of the cell containing the last segment of the PL-path before reaching the end-point, in other words, $\pi(k-1) \widetilde{\succeq} \pi(k)$. Figure 6 illustrates this situation in both path x and path y for different situations.

We are now ready to define the formal semantics of SLCS on $\mathcal{M}(\mathcal{X})$.

Definition 4.3 (SLCS semantics on $\mathcal{M}(\mathcal{X})$). Consider $\mathcal{M}(\mathcal{X}) = \left\langle \widetilde{K}, \widetilde{\preceq}, \widetilde{V} \right\rangle$. Given $\widetilde{\sigma} \in \widetilde{K}$, satisfaction $\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \phi$ over formulas ϕ is given by the following inductive clauses, where we let $[\![\phi]\!]^{\mathcal{M}}$ denote the set $\{\widetilde{\sigma} \in \widetilde{K} \mid \mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \phi\}$:

$$
\begin{aligned}
&\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \top && \text{always holds} \\
&\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash p && \Longleftrightarrow && \widetilde{\sigma} \in \widetilde{V}(p) \text{ for } p \in \text{AP} \\
&\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \neg\phi && \Longleftrightarrow && \mathcal{M}(\mathcal{X}), \widetilde{\sigma} \nvDash \phi \\
&\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \phi \wedge \psi && \Longleftrightarrow && \mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \phi \text{ and } \mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \psi \\
&\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \Box\phi && \Longleftrightarrow && \forall \widetilde{\tau} \in \widetilde{K}. \text{ if } \widetilde{\sigma} \widetilde{\preceq} \widetilde{\tau} \text{ then } \mathcal{M}(\mathcal{X}), \widetilde{\tau} \vDash \phi \\
&\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \gamma(\phi, \psi) && \Longleftrightarrow && \text{there exists a } \pm\text{-path } \pi : \{0, \ldots, k\} \xrightarrow{\pm} \widetilde{K} \text{ such that} \\
& && && \pi(0) = \widetilde{\sigma}, \pi(k) \in [\![\psi]\!]^{\mathcal{M}} \text{ and } \pi(\{1, \ldots, k-1\}) \subseteq [\![\phi]\!]^{\mathcal{M}}
\end{aligned}
$$

The clauses for the Boolean operators and for $\square$ are the standard interpretation of modal formulas on Kripke models (and on topological spaces, via the Alexandrov topology, see [BB07]). In the semantic clause for γ we use $\pm$-paths since, as previously pointed out, these paths naturally correspond to PL-paths and Lemma 3.11 allows us to restrict our attention to this class of topological paths. We can give an example of this correspondence based on Figure 8: the PL-path π' (on the left) witnesses that $x \in [\![\gamma(r, \neg(r \vee g))]\!]^{\mathcal{X}}$, and the corresponding $\pm$-path π (on the right) witnesses that $\widetilde{AB} \in [\![\gamma(r, \neg(r \vee g))]\!]^{\mathcal{M}}$.

The following theorem shows that this correspondence holds for every formula of the logic.

Theorem 4.4. *Let $\mathcal{X} = \langle P, \mathrm{K}, V \rangle$ be a polyhedral model and x a point of P. Let $\sigma \in \mathrm{K}$ be the unique simplex such that $x \in \widetilde{\sigma}$. For every formula ϕ of SLCS we have $\mathcal{X}, x \vDash \phi \iff \mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \phi$.*

4.1. **Geometric Model-checking Algorithm.** We briefly present the main aspects of the geometric model checking algorithm for SLCS over polyhedra. The model checking algorithm takes as inputs a Kripke model $\mathcal{M}(\mathcal{X})$ of polyhedron $\mathcal{X}$ and an SLCS formula ϕ. The output is the satisfaction set $\mathrm{Sat}(\phi) = \{\widetilde{\sigma} \in \widetilde{\mathrm{K}} \,|\, \mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \phi\} = [\![\phi]\!]^{\mathcal{M}}$ of nodes in the model $\mathcal{M}(\mathcal{X})$ that correspond to the set of cells in $\mathcal{X}$ that satisfy formula ϕ, i.e., the algorithm is a *global* model checking algorithm. The satisfaction set Sat is defined recursively on the structure of SLCS formulas in the usual way (see for example [BK08]). The algorithms for the Boolean operators are straightforward, and thus omitted. The algorithm for the $\square$ operator takes a set $\mathrm{Sat}(\phi) = [\![\phi]\!]^{\mathcal{M}}$ and computes the corresponding satisfaction set $\mathrm{Sat}(\square\phi) = [\![\square\phi]\!]^{\mathcal{M}} = \{\widetilde{\sigma} \in \mathrm{Sat}(\phi) \,|\, \mathtt{out}(\widetilde{\sigma}) \subseteq \mathrm{Sat}(\phi)\}$, where $\mathtt{out}(\widetilde{\sigma})$ is the out-neighbourhood of $\widetilde{\sigma}$, i.e., $\mathtt{out}(\widetilde{\sigma}) = \{\widetilde{\tau} \,|\, \widetilde{\sigma} \preceq \widetilde{\tau}\}$. We omit a detailed description of the algorithm, since it only involves basic set-theoretic operations. We give a more detailed description of the algorithm for the spatial reachability operator γ that, given $\mathrm{Sat}(\phi) = [\![\phi]\!]^{\mathcal{M}}$ and $\mathrm{Sat}(\psi) = [\![\psi]\!]^{\mathcal{M}}$, computes the satisfaction set $\mathrm{Sat}(\gamma(\phi, \psi)) = [\![\gamma(\phi, \psi)]\!]^{\mathcal{M}}$.

The semantics of γ, as of Definition 4.3, is computed via a variant of the flooding procedure that was already successfully employed in our previous work on point-based spatial model checking algorithms [CLLM16a, CGL$^+$15, Gri16], retaining its asymptotic complexity (in the geometric case linear in the number of nodes in $\mathcal{M}(\mathcal{X})$).

The pseudo-code with comments is reported in Algorithm 4.5. In the code, we make use of the additional operators on cells $\mathtt{out}$ (previously introduced) and $\mathtt{in}$, the in-neighbourhood, defined as $\mathtt{in}(\widetilde{\sigma}) = \{\widetilde{\tau} \,|\, \widetilde{\sigma} \succeq \widetilde{\tau}\}$. Both operators are lifted to sets in the obvious way. Therein, for brevity, we call a $\pm$-path π a "good" path if it is witnessing the formula $\gamma(\phi, \psi)$, i.e., the path must map $(0,1)$ to $[\![\phi]\!]^{\mathcal{M}}$ and $\{1\}$ to $[\![\psi]\!]^{\mathcal{M}}$. Figure 9 shows a step-by-step example of an execution of the flooding algorithm described in Algorithm 4.5 for the property $\gamma(r, g)$, where r denoted red and g denotes green, in the polyhedral model and corresponding Kripke model shown in Figure 9a. Recall that the definition of the reachability operator γ states that the first point of the paths passing by red and reaching green does not need to be red itself, which explains why also point B is included in the results.

We include a concise proof of the correctness of the algorithm.

Correctness, sketch. To be consistent with the comments in the pseudo-code, we keep calling a $\pm$-path $\pi : \{0, \dots, k\} \to \widetilde{\mathrm{K}}$ witnessing the satisfaction of $\gamma(\phi, \psi)$ a "good" path.

Algorithm 4.5. Pseudo-code for model checking the spatial reachability operator.

```
1      Input:   [[φ]]^M, [[ψ]]^M sets of nodes of M(X) satisfying φ resp. ψ
2      Output:  [[γ(φ,ψ)]]^M satisfaction set of nodes in M(X)
3
4      // frontier: the points queued for the next iteration.
5      // Initialized with the next to last points of a good path.
6      let frontier = [[φ]]^M ∩ out([[ψ]]^M)
7
8      // flooded: all the points that are in the middle of a good path.
9      let flooded = frontier
10
11     while frontier ≠ ∅:
12         let σ = frontier.pop()
13         for every τ ∈ in(σ) ∪ out(σ):
14             if τ ∉ flooded and τ ∈ [[φ]]^M:
15                 frontier.add(τ)
16                 flooded.add(τ)
17
18     // result: the starting points of a good path.
19     let result = in(flooded)
20
21     return result
```

First, notice that we can divide a good path into three parts: the initial point $\pi(0)$, the points on the central segment $\pi(\{1, \ldots, k-1\})$ all satisfying ϕ and the final point $\pi(k)$ satisfying ψ. To compute the set $[[\gamma(\phi,\psi)]]^{\mathcal{M}}$ we work "backwards" from an initial set of nodes that satisfy ϕ and that have an outgoing edge to a node satisfying ψ. So, first, we compute the set $C := [[\phi]]^{\mathcal{M}} \cap \text{out}([[\psi]]^{\mathcal{M}})$ (the cells corresponding to nodes with index $\pi(k-1)$ on some good path π). Then we use a standard flooding procedure to build the set D of the nodes of the graph that are connected to C via a non-directed path, i.e., abstracting from the direction of the edges, passing only through nodes satisfying $[[\phi]]^{\mathcal{M}}$ (these correspond to the cells of the form $\pi(j)$, with $j = 1, \ldots, k-1$, for any good path π in the polyhedron). Finally we compute the set $[[\gamma(\phi,\psi)]]^{\mathcal{M}} = \text{in}(D)$ (the cells corresponding to the initial nodes of good paths, i.e., $\pi(0)$ for any good path π). ☐

4.2. **Computational Complexity.** In what follows, we indicate by n the number of cells of $\widetilde{K}$ and by d the dimension of K, that is, the *maximum* dimension of a simplex σ in K. Therefore, the number of nodes of the encoding $\mathcal{M}(X)$ is n. Each simplex σ having dimension $d_\sigma \leq d$ has $d_\sigma + 1$ vertices. Furthermore, in the graph encoding $\widetilde{K}$, the in-neighbourhood of each $\widetilde{\sigma} \in \widetilde{K}$ is precisely the set of the proper faces of σ, having cardinality $2^{d_\sigma+1} - 1$. By this, the total number of edges of $\mathcal{M}(X)$ is at most $n \cdot (2^{d+1} - 1)$. We let N to be the size of this encoding, that is, the sum of the number of nodes and edges, which is at most equal to $n + n \cdot (2^{d+1} - 1) = n \cdot 2^{d+1}$.

Note that the complexity of our encoding grows exponentially in d if d is not fixed. The design space for algorithms that scale better with d, possibly exploiting specialised data structures (see, e.g., the recent work [BM14]) will be explored in future research, depending

(A) A polyhedral model $\mathcal{X}$ and its corresponding Kripke model $\mathcal{M}(\mathcal{X})$.

frontier $= \{AC, BC, ABC\}$

(B) Initialization of frontier (lines 4-5).

frontier $= \{A, AB\}$
flooded $= \{AC, BC, ABC\}$

(C) Initialization of flooded and first flooding iteration (lines 7-15).

frontier $= \emptyset$
flooded $= \{A, AB, AC, BC, ABC\}$

(D) Second flooding iteration (lines 10-15).

flooded $= \{A, AB, AC, BC, ABC\}$
result $= \{A, B, C, AB,$
$\qquad\qquad AC, BC, ABC\}$

(E) Computation of result (lines 17-18).

Figure 9: An example of application of the algorithm described in Algorithm 4.5. Subfigure 9a depicts a polyhedral model $\mathcal{X}$ (on the left) and its corresponding Kripke model $\mathcal{M}(\mathcal{X})$. Subfigures from 9b to 9e depict the steps performed on $\mathcal{M}(\mathcal{X})$ to compute the satisfaction set $\mathrm{Sat}(\gamma(r, g))$, with reference to the pseudo-code in Algorithm 4.5. For each subfigure the value of the relevant variables at the end of the computation step are indicated and illustrated in corresponding colours in both models (blue for frontier, orange for flooded and purple for result).

on the considered use case. In the case under consideration (i.e., 3D meshes) we have the fixed dimension $d = 3$. Therefore the "exponential" contribution of d to the computational complexity becomes a constant. In other words, when the dimension d is fixed, as in 3D meshes, the size of $\mathcal{M}(\mathcal{X})$ is of order $\mathcal{O}(n)$.

Note that the flooding procedure (Algorithm 4.5) has linear computational complexity in the number of nodes and edges of $\mathcal{M}(\mathcal{X})$, that is, N. The computation of the Boolean operators and of the $\square$-operator are also linear in N. As in [CLLM16a], since each subformula is checked independently from the others, the asymptotic computational complexity of the model checking algorithm is of order $\mathcal{O}(N \cdot h)$, where h is the cardinality of the set of subformulas of the SLCS formula to be checked.

As remarked before, **once the dimension d of the image is fixed** the exponential contribution of d to the computational complexity is negligible and therefore the total complexity of the spatial model checking algorithm is of order $\mathcal{O}(n \cdot h)$.

Finally, we briefly address the complexity of the encoding. In the current prototype implementation of the model-checking algorithm (see Section 5 for further details), the input polyhedron is described by a list of n simplexes with maximum dimension d, each one being represented by a list of vertices. To compute the Kripke frame of Definition 4.1 from this description, an explicit enumeration is performed of the subsets of each simplex, incrementally building the arrays of out- and in-neighbourhoods. This results in a time complexity in $\mathcal{O}(N)$, which becomes $\mathcal{O}(n)$ once the dimension d is fixed. Therefore, for d fixed, the total complexity (encoding plus model checking) is in $\mathcal{O}(n \cdot h)$.

5. PolyLogicA: a Model Checker for Polyhedra

Based on the theory and the model checking algorithm presented in Section 4, we developed the prototype model checker PolyLogicA: a *Polyhedral Logic-based Analysis tool*. The model checker is implemented in the functional language FSharp.[8] PolyLogicA is Free and Open Source Software, distributed under the Apache 2.0 license.[9]

Currently PolyLogicA represents a polyhedral model $\mathcal{X}$ through an explicit encoding of the Kripke model $\mathcal{M}(\mathcal{X})$ according to Definition 4.1. The encoding is stored as a graph having the cells as nodes and with the covering relation of $\widetilde{\preceq}$ as the edge relation. The current implementation stores the out-neighbourhood and the in-neighbourhood of each node σ in two separate arrays, allowing access in constant time to these sets.

5.1. **Functional description.** A PolyLogicA specification consists of a text file that can make use of four commands: `let`, for declaring functions and constants; `import`, for importing libraries of such declarations; `load`, to specify the file to be loaded as a model; `save`, to specify the logic formulas that need to be computed, and saved, possibly making use of previous `let` declarations.

Models are required to be based on a fixed simplicial complex. The given simplicial complex constitutes the subspace of $\mathbb{R}^d$ that is explored by the tool. Thus, one needs to explicitly include, in the input model, the "environment" in which the objects live. As an example, consider the simplicial complex depicted in Figure 4. The semantics of the formula

---

[8]See https://www.fsharp.org

[9]The tool is currently available in a branch of the main VoxLogicA repository, see https://github.com/vincenzoml/VoxLogicA.

$\phi = g \lor r \lor b$ is indeed the set of points S that belong to the whole coloured triangle (made up of four smaller ones).

The file describing the input model uses a custom `json`-based[10] format. The information contained in the file consists of: a list p of d-dimensional vectors, denoting the coordinates of the 0-cells of the polyhedron; a list of atomic proposition identifiers; a list of simplexes. Each simplex is specified by the list of the indexes of its vertices in p, and its specification also contains the list of atomic propositions holding at the cell corresponding to the simplex.

Logic formulas are just a concrete syntax for SLCS as presented in Section 3. Currently, PolyLogicA does not implement additional extra-logical operators (contrary to VoxLogicA, which also implements imaging primitives). PolyLogicA is in spirit a *global, explicit-state* model checker, that is, the set of simplexes satisfying a given formula is computed and returned at once. The output of the model checker is a list, in `json` format, containing an element for each formula ϕ that the specification requires to be checked. Each element of such list contains in turn a list representing the truth values of ϕ at each point (cell) of the input model. Finally, a simple 3D, web-based visualizer has been implemented along with the prototype (see the screenshot in Figure 5.4.2), which will be refined in future work.

5.2. **Implementation Details.** The current version of PolyLogicA has been implemented sharing part of the code base for point-based spatial model checking with VoxLogicA. PolyLogicA inherits from its parent tool the multithreaded, memoizing computation engine, and the parser for the input language of the tool. Basically, after expanding `let` bindings, each formula is converted into a *directed acyclic graph* where nodes are *tasks*, and arcs are *dependencies*. Each task is a basic logical primitive, to be applied to specific arguments. Task A depends upon task B if and only if the result of B is an argument of A. The implementation guarantees that, while being constructed, such data structure is kept *minimal* in the sense that the same primitive on the same arguments will never be added twice to the graph, thus implementing *memoization* at a syntactic level, without the need to hash possibly large sets of states at run-time. After having been constructed, the task graph is executed in parallel as much as possible, exploiting the available CPU cores.

5.3. **A Visualiser for PolyLogicA.** In addition to the polyhedra model checker we developed PolyVisualiser, a prototype polyhedra visualiser. The user interface of the visualiser is shown in the screenshots in Figure 10. The visualiser is an interactive tool with which polyhedra can be explored from different perspectives by means of the mouse or an other pointing device. The tool allows to zoom in and out on the visualised model, to translate it rigidly and to turn it around a fixed point. The main purpose of the visualiser is to inspect the result of spatial properties that were checked using PolyLogicA. From a pop-up menu, shown on the right of the screenshot, a property can be selected and from further menu items one can select the way in which the results of the evaluation of the property will be visualised. For example, in the screenshot in Figure 10 the cells that satisfy the selected property are shown in green and the cells that do *not* satisfy the property are shown in red. Also the degree of transparency of the colours can be set, thus facilitating the inspection of the interior of 3D objects. Cells that are points or line segments or triangles are coloured in a direct way. To facilitate inspection of the solid components, being them covered by triangles, tetrahedra are reduced in size and coloured without transparency. Moreover, the

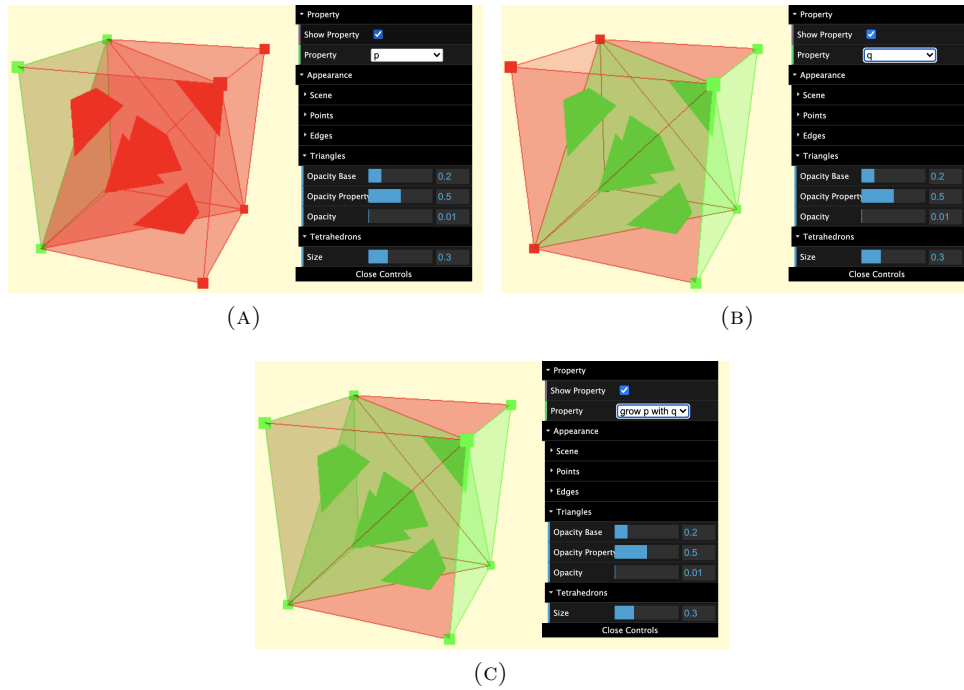---

[10]See `https://www.json.org/`.

Figure 10: Visualiser for PolyLogicA: Model of a simple cube showing cells satisfying the atomic propositions "p" and "q" in green ((10a) and (10b), respectively) and model checking result (in green) of cells satisfying "grow p with q" (10c).

relative size of tetrahedra can be manipulated real-time through the appropriate control panel.

In the following sections PolyVisualiser is used to illustrate two examples of realistic size. In those examples, rather than using the colour green for cells in the polyhedron that satisfy the selected formula and the colour red for those cells that do not satisfy the formula, we will use different levels of transparency while retaining the original colour of the cells. Cells that satisfy the selected property are shown in their original colour but made less transparent (i.e., more opaque) than cells that do not satisfy the property. This way the contours of the complete model are preserved and may serve as a visual reference framework for the interpretation of the 3D spatial model checking results which pop-out as opaque objects as shown in Figure 14.

5.4. **Proof of Concept Experiments.** In order to obtain a first impression of the actual feasibility of our (first-in-its-kind) geometric model checking approach we present two examples: one involving the analysis of a 3D maze (generated by the authors) and one involving the analysis of an existing mesh model of the anatomy of the human body. Both examples are analysed with PolyLogicA and we show screenshots of the PolyVisualiser to illustrate the results. The results are also made available online[11] for inspection in an interactive way.

---

[11]http://ggrilletti2.scienceontheweb.net/polyVisualizer/polyVisualizer_static_maze.html

5.4.1. *A 3D Maze.* The first example concerns a 3D maze.[12] The maze, shown in Figure 13a, consists of "rooms" that are connected by "corridors". The rooms come in four colours: white, black, green and one room in red. The green rooms are all situated at the outer boundary of the maze and represent the surroundings of the maze that can be reached via an exit. The white, black and red rooms, and related corridors, are situated inside the cube and form the maze itself. Figure 13b shows all the white and black rooms. Figure 13c shows the red room situated at the centre of the maze. All corridors between rooms are dark grey. The valid paths through the maze should only pass by white rooms (and related corridors) to reach a green room without passing by black rooms or corridors that connect to black rooms.

Typical spatial queries or properties that are of interest for such a maze are:

Q1: Which are the white rooms and connecting corridors from which one can reach a green room (i.e., an exit) without having to pass by a black room? Show those white rooms, the related corridors and the green exit.

Q2: Which are the white rooms and connecting corridors from which both a green room (exit) and a red room can be reached, without having to pass by black rooms (and related corridors)? Show those white rooms, the related corridors, the green exit and the red room.

Q3: Which are the white rooms (and related corridors) from which it is *not* possible to reach a green room without having to pass by a black room?

Figure 12 shows the specification of the above properties in a syntax close to the spatial logic introduced in Section 3. This syntax is shared with that of VoxLogicA (see [CBLM21] for a tutorial) and is mostly self-explanatory. The boolean operators are | (or), & (and), ! (not). The spatial reachability operator γ is denoted by the keyword `through`. The surround operator `sur` is implicitly defined in terms of the operator `through` according to its definition provided in Section 3. In the first line of Figure 12, the model file is loaded. In lines 4 to 7 atomic propositions are defined for the simplicial cells of the various coloured rooms. In line 8, the atomic proposition of cells belonging to corridors is defined. These definitions make direct reference to the information stored in the elements of the model file, in `json` format, representing the cells of the polyhedron model. In particular, `ap` denotes an atomic proposition associated with such elements.

In line 11, a simple application of these atomic propositions is defined. The property `blackOrWhite` is satisfied by all cells of black rooms and those of white rooms. The result of this property can be saved (as shown in line 44) and used by PolyVisualiser to visualise the outcome. The saved file essentially consists of a list of booleans indicating for each cell whether it satisfies the property. The result is shown in a screenshot of the visualiser in Figure 13b where the cells that satisfy the property are opaque and all the others almost transparent. Similarly, in Figure 13c the cells that satisfy the atomic proposition `red` are shown.

Lines 14 to 20 define properties to distinguish different kinds of corridors. In particular, corridors that connect white rooms (`corridorWW`), those between white and green rooms (`corridorWG`), those between white and red rooms (`corridorWR`) and those between white and black rooms (`corridorWB`). These will turn useful in the specification of the three properties Q1, Q2 and Q3. A point satisfies `corridorWW` if it (lays in a cell that) belongs to a corridor (i.e., a cell satisfying `corridor`) and from which only white rooms can be reached

---

[12]The maze has been generated by the authors.

$$
\begin{aligned}
\text{corridorWW} &\equiv \gamma(C, W) \wedge \neg\gamma(C, G \vee B \vee R) \\
\text{corridorWG} &\equiv \gamma(C, W) \wedge \gamma(C, G) \\
\text{corridorWR} &\equiv \gamma(C, W) \wedge \gamma(C, R) \\
\text{corridorWB} &\equiv \gamma(C, W) \wedge \gamma(C, B) \\
\text{whiteToGreen} &\equiv \gamma((W \vee \text{corridorWW} \vee \text{corridorWG}), G) \\
\text{Q1} &\equiv \text{whiteToGreen} \vee \gamma(G, \text{whiteToGreen}) \\
\text{Q2} &\equiv \gamma((\text{Q1} \vee \text{corridorWR}), R) \vee \gamma((R \vee \text{corridorWR}), \text{Q1}) \\
\text{Q3} &\equiv (W \vee \text{corridorWW}) \wedge \neg\text{whiteToGreen}.
\end{aligned}
$$

Figure 11: SLCS formulas expressing properties Q1, Q2 and Q3; atomic proposition letters
G, W, B, R, C are assumed given and their meaning is the obvious one (C for
"corridor", G for green and similarly for the other colours).

via the corridor itself (i.e., only cells satisfying `white` but neither `green`, nor `black`, nor `red`
can be reached from cells of the corridor).

In lines 24 and 27 property Q1 is specified as `whiteToGreen`. It is expressed in terms of
the `through`-operator (γ). It says that we want all (cells of) white rooms, corridors between
white rooms and corridors between white and green rooms by which one can reach a green
room, i.e., without passing by black rooms or corridors connected to black rooms. If we
also want to include the green room that is being reached in the set of results we have to
add it, as is done in line 27 in `connWG`, leading to all cells satisfying `whiteToGreen` and
the (cells of) green rooms from which cells satisfying `whiteToGreen` can be reached. Cells
satisfying `connWG` are shown in their original opaque colour in Figure 14a. Cells that do
not satisfy the property are shown in a transparent manner. Note that, in general, the
`through(x,y)`-operator would also admit satisfaction by cells that are adjacent to those
satisfying `x` in that case, as shown in the example of Figure 5b. In the model of the maze
rooms and corridors do not have such adjacent cells. However, in general one could consider
the use of `(x | y) & through(x,y)` to make sure that only cells satisfying at least `x` or `y`
are part of the result. The choice for the particular definition of `through` (and γ) has been
motivated by the fact that it poses minimal restrictions. More restricted variants can be
easily defined based on `through`. This would not be the case if one opts for a less basic
definition of `through`.

In line 30 property Q2 is specified as `connRWG`. In this case we are looking for white
rooms and related corridors from which both a green room and a red one can be reached,
and we also want to include in the result such red and green rooms. The specification
makes use of `connWG` and allows passing by corridors connecting white rooms to red rooms
`corridorWR`. The result is shown in Figure 14b.

In line 34 property Q3 is specified as `whiteNoGreen`. In this case we are looking for
white rooms and related corridors from which it is *not* possible to reach a green room without
passing by a black room. This can be specified as white rooms and corridors between white
rooms that are not satisfying `whiteToGreen` (i.e., property Q1 above). The same property
could also be specified in an alternative way making use of the surround operator `sur` defined
in lines 37-38. In that case these cells are characterised as those belonging to white rooms
and corridors between white rooms that are completely surrounded by corridors leading only
to black rooms in specification `whiteSblack` in line 41. The result is shown in Figure 14c.

	Number of tasks	Check (ms.)	Total time (ms.)
Property Q1 (connWG)	24	200	5,450
Property Q2 (connRWG)	31	250	5,500
Property Q3 (whiteNoGreen)	25	100	5,350
All properties in one run	34	300	5,550

Table 1: PolyLogicA performance results (rounded to the nearest multiple of 50ms) of the evaluation of the specification in Figure 12 on three properties of the 3D mesh of the maze shown in Figure 13. The size of the model is 147,245 cells. The time spent in parsing the input file (circa 12 megabytes) is about 4 seconds, whereas building the Kripke structure takes about 1 second. If all properties are evaluated in the same specification, the execution time is just slightly above the maximum one, since the machine has 8 physical cores, and PolyLogicA evaluates independent properties in parallel needing less than 8 cores for the maze example.

Finally, in line 44 an example is shown of saving a particular result, in this case that of property blackOrWhite. Saving the other results can be performed in a similar manner and is not shown in the specification.

For completeness, in Figure 11, the SLCS formulas expressing properties Q1, Q2 and Q3 are shown.

Table 1 shows information on the model checking time for the properties Q1-Q3 individually and when evaluated all together. Also the total time is shown that includes the parsing of the model, the generation of the Kripke model and the actual model checking time. It is easy to see that most of the time goes in preparing the model (4 seconds for parsing and 1 second for building the Kripke model) rather than the actual model checking which only takes a few hundred milliseconds. This is very encouraging, because it means that the actual model checking procedure is not the bottleneck and we are confident that more efficient parsing and Kripke model generating procedures can be found than the non-optimised ones we used for this first proof-of-concept set-up. Work on such optimisations is planned as part of future work. Note also that it is possible to verify several properties at once in which case the Kripke structure needs to be generated only one time. In future work an option could be to enable saving and loading such Kripke structures once they have been generated in order to reduce the total time needed for analysis. The evaluation was performed on a desktop machine equipped with an Intel core i9 9900k cpu and 32Gb of RAM.

Even though the 3D maze example seems simple, the fact that examples like this can be generated in various forms makes it a promising example for a future benchmark. The example lends itself for endless variations in size and shape of the maze, introducing "holes" or forms of "rings". Setting up a suitable benchmark for geometric model checking of polyhedra is planned as part of future work.

5.4.2. *Digital Anatomy.* In the second example we apply PolyLogicA on an *existing* 3D mesh from the medical domain with the aim to assess the feasibility of the geometric model checking approach on existing meshes of realistic size. The mesh visualised in Figure 16(a) consists of about 1.5 milion simplicial cells. A custom converter has been implemented to

```
1  load model = "mazeModel.json"
2
3  // Atomic propositions for rooms and corridors
4  let green      = ap("G")
5  let white      = ap("W")
6  let black      = ap("B")
7  let red        = ap("R")
8  let corridor   = ap("corridor")
9
10 // Black or White rooms
11 let blackOrWhite = black | white
12
13 // Corridors: white-to-white, white-to-green, white-to-red, white-to-
       ↪ black:
14 let corridorWW = through( corridor, white ) &
15                  !through( corridor, green | black | red )
16 let corridorWG = through( corridor, white ) &
17                  through( corridor, green )
18 let corridorWR = through( corridor, white ) &
19                  through( corridor, red )
20 let corridorWB = through( corridor, white ) &
21                  through( corridor, black )
22
23 // Q1: White rooms from which a green room can be reached not passing by
       ↪ black rooms
24 let whiteToGreen = through((white | corridorWW | corridorWG), green)
25
26 // as Q1 but including the green room that is reached
27 let connWG = whiteToGreen | through(green,whiteToGreen)
28
29 // Q2: White rooms from which both a red and a green room can be reached
       ↪ not passing by black rooms
30 let connRWG = through((connWG | corridorWR), red) |
31                  through((red | corridorWR), connWG)
32
33 // Q3: White rooms with no path to green rooms and their connecting
       ↪ corridors
34 let whiteNoGreen = (white | corridorWW) & !whiteToGreen
35
36 // Surround operator in terms of reach in the polyhedra setting
37 let reach(x,y) = x | through(y,x)
38 let sur(x,y)   = x & !reach(!(x | y),!y)
39
40 // Q3(alternative): White rooms and their connecting corridors surrounded
       ↪  only by corridors to black rooms
41 let whiteSblack = sur((white | corridorWW), corridorWB)
42
43 // Save the result for property blackOrWhite
44 save "blackOrWhite" blackOrWhite
```

Figure 12:   PolyLogicA specification of some spatial properties for the 3D maze model.
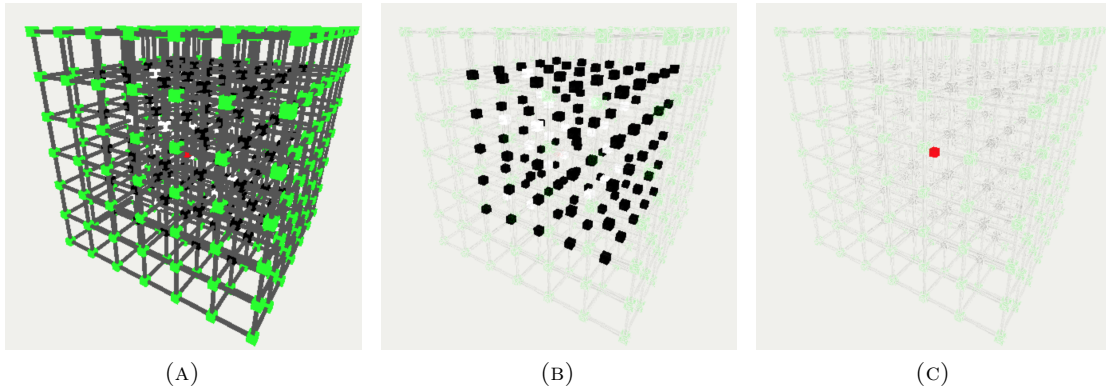
(A)   (B)   (C)

Figure 13:   3D maze (13a), black and white (13b) and red rooms (13c) in the 3D maze.

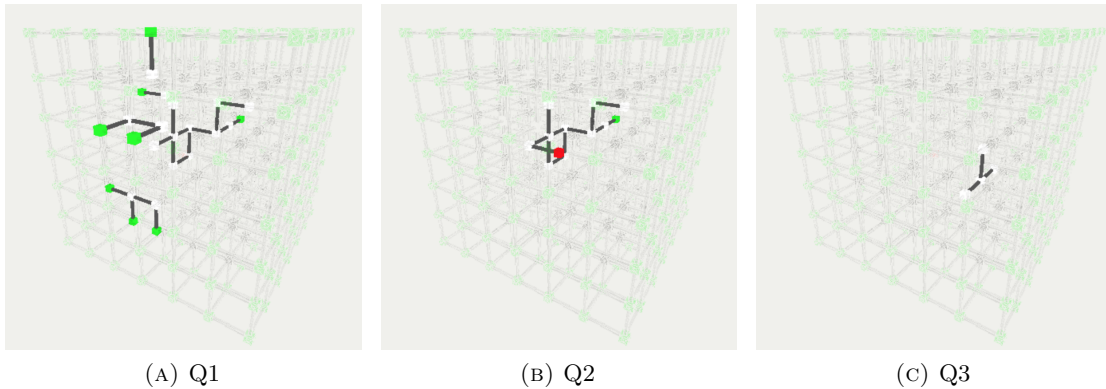(A) Q1   (B) Q2   (C) Q3

Figure 14: Spatial model checking results of the properties in Figure 12 for the 3D maze
of Figure 13. Q1: White rooms and their connecting corridors from which a
green room can be reached not passing by black rooms, including the green room
that is reached (`connWG`); Q2: White rooms and their connecting corridors from
which both a red and a green room can be reached not passing by black rooms
(`connRWG`); Q3: White rooms and their connecting corridors with no path to
green rooms (`whiteNoGreen` or equivalently `whiteSblack`).

obtain a model file in `json` format from the input `obj` mesh;[13] the atomic propositions used
are strings of the form `"Ci"` for `C` in $\{r, g, b\}$ (standing for *red, green, blue*) and `i` an integer
between 0 and 3. Each such atomic proposition denotes the intensity level at each point of
the red, green or blue component of the colour vector associated to the simplicial cell, with
intensities quantised to four discrete target levels indicated by the integers.

Based on such atomic propositions, it is possible to encode spatial properties involving
approximations of the colour gradients[14] that are visible in the rendered mesh. Using the
specification of spatial properties shown in Figure 15, we have checked the properties `heart`

---

[13] *Wavefront `obj`* is a widely used file format for 3D meshes. See `https://en.wikipedia.org/wiki/`
`Wavefront_.obj_file`

[14] This type of mesh has colours associated to vertices, and no textures. In future versions of the tool we
plan to encode colour intervals in the logic, in a similar way as has been done in the parent tool VoxLogicA.

(Figure 16(b)), `spleen` (Figure 16(c)), `vein` (Figure 16(d)), and `selectedVein` (Figure 16(e)). The syntax of the specification is the same as the one explained in the 3D maze example. Some definitions (e.g., `spleenWithError`) need to handle errors in the segmentations due to the usage of quantised colours leading to somewhat loose approximations; this is no problem for our purposes, as in this example we merely want to illustrate a first feasibility test and performance of the tool on existing meshes. After loading the model file (line 1), in the specification first some areas are identified based on their approximative colour (lines 3–8). In particular, a blueish colour is defined as a combination of the blue and red intensity levels. The core of the spleen can be found in a similar way, as well as an over-approximation of the spleen by a relaxation of the constraint on the level of red. Heart, spleen or kidneys are identified by a particular combination of the levels of red and green.

This simple first approximation is then used to identify the individual organs exploiting, for example, the knowledge that veins are blueish and leading to the core of the spleen (line 10, definition `vein`). The spleen itself is first approximated by `spleenOver`, an over-approximation of the spleen that touches the veins, but may have some overlap with the heart. Recall here that the `through`-operator may be satisfied by some points (cells) that do not satisfy `spleenOver` (see the example in Figure 5b).

The heart is specified as that part of `heartOrSpleenOrKidneys` that is not part of the over-approximated spleen that can reach a vein `spleenWithError`, but can reach a vein by itself. Subsequently, the spleen can be characterised more precisely by excluding the points belonging to the heart (line 12). Finally, the reachability operator `through` is used (line 14) to identify the veins that reach the spleen but not the heart.

We have tested the specification using PolyLogicA on a desktop machine equipped with an Intel core i9 9900k cpu and 32Gb of RAM.[15] Three models have been used: the original one, and two versions that have been obtained by simplifying the original mesh, using a built-in algorithm in MeshLab [CCC+08]. The number of tasks executed (the nodes of the directed acyclic graph described in Section 5.2, that is, the cardinality of the set of subformulas) is 33. Table 2 reports the model size and the execution time, broken down into parsing of the `json` model file, computing the Kripke structure, and actual model checking. Note that parsing a terse textual format for such large objects is time consuming, and ought to be replaced in future work by functions to load and save more specific mesh-based file formats. However, in the meantime, we find these results particularly encouraging, as the model checking times are quite small, and would permit to check many more formulas in a single run, thus compensating for the time needed to generate the Kripke structure.

Future work will include the implementation of a fast loader for 3D meshes, in order to eliminate the parsing of (very large) model files in `json` format, and the optimisation of the translation from simplicial complexes to Kripke models, which currently exploits purely-functional data structures for ease of prototyping. We note in passing that the intermediate Kripke model may be cached, for speeding up the execution of multiple analyses on the same model.

Future work will also include further experiments with more complex spatial formulas and different kinds of meshes with the aim to generate a basic benchmark suitable for future comparisons of performance.

However, this is ongoing work, as the presence or colours, textures, materials, and so on, may depend upon the chosen file format of the mesh and will require further investigation.

[15]The memory limit is actually never hit, even with lower amounts of available RAM on other machines.

	Size (cells)	**Parse** (secs.)	**Kripke** (secs.)	**Check** (secs.)	**Total** (secs.)
Small	91,771	2.5	0.5	0.3	3.3
Medium	591,269	15	7	1.6	23.6
Full-size	1,587,430	44	32	5	81

Table 2: PolyLogicA performance results of the evaluation of the specification in Figure 15 on three variants of the 3D Medical mesh shown in Figure 16a.

```
1   load model = "RAJA2.json"
2
3   let blueish = (ap("b3")|ap("b2")) & (ap("r0")| ap("r1"))
4   let cyan = ap("r2") & (ap("g0"))
5
6   let spleenCore = ap("b3") & (ap("r1")) & ap("g1")
7   let spleenOver = ap("b3") & (ap("r2")|ap("r1")) & ap("g1")
8   let heartOrSpleenOrKidneys = ap("r2") & (ap("g1"))
9
10  let vein = through(blueish,spleenCore)
11  let spleenWithError = (spleenOver | through(spleenOver,vein))
12  let heart = through(heartOrSpleenOrKidneys & (not(spleenWithError)),vein)
13  let spleen = spleenWithError & (!heart)
14  let selectedVein = vein & (!through(vein,heart))
```

Figure 15: Specification of the spatial properties to produce the results shown in Figure 16.

6. Simplicial Bisimilarity

A further, more theoretically inspired, direction of research aiming at increasing the performance of geometric spatial model checking is to exploit suitable spatial bisimilarity to reduce the models. The main idea is to identify areas that satisfy the same spatial properties, before applying model checking. Similar approaches have been exploited in traditional model checking techniques, based on bisimilarity for modal logics (see e.g. [Ben84]).

As a first step into this direction, in this section we characterise logical SLCS equivalence via *simplicial bisimilarity*. Recall the results summarised in [APB07b], defining bisimilarity for topological spatial logics, so that any two points are bisimilar if and only if they are logically equivalent.

To account for the addition of the reachability operator in the logical language, the definition of bisimilarity makes use of the point-wise lifting of a relation to a path, defined in a formal way below. In the following, fix a model $\mathcal{X} := \langle P, \mathrm{K}, V \rangle$.

Definition 6.1. Given a relation $R \subseteq P \times P$, let the *extension of R to paths* be the binary relation between paths $\hat{R}$, such that $\pi_1 \hat{R} \pi_2$ if and only for all $t \in [0, 1]$ we have $\pi_1(t) R \pi_2(t)$.

Definition 6.2 (Simplicial bisimilarity). A binary relation $\sim \subseteq P \times P$ is a *simplicial bisimulation* if and only if for all x, y with $x \sim y$:

(1) for all $p \in \mathrm{AP}$, $x \in V(p) \iff y \in V(p)$;

(A)
3D medical image

(B)
Heart (red)

(C)
Spleen (brown)
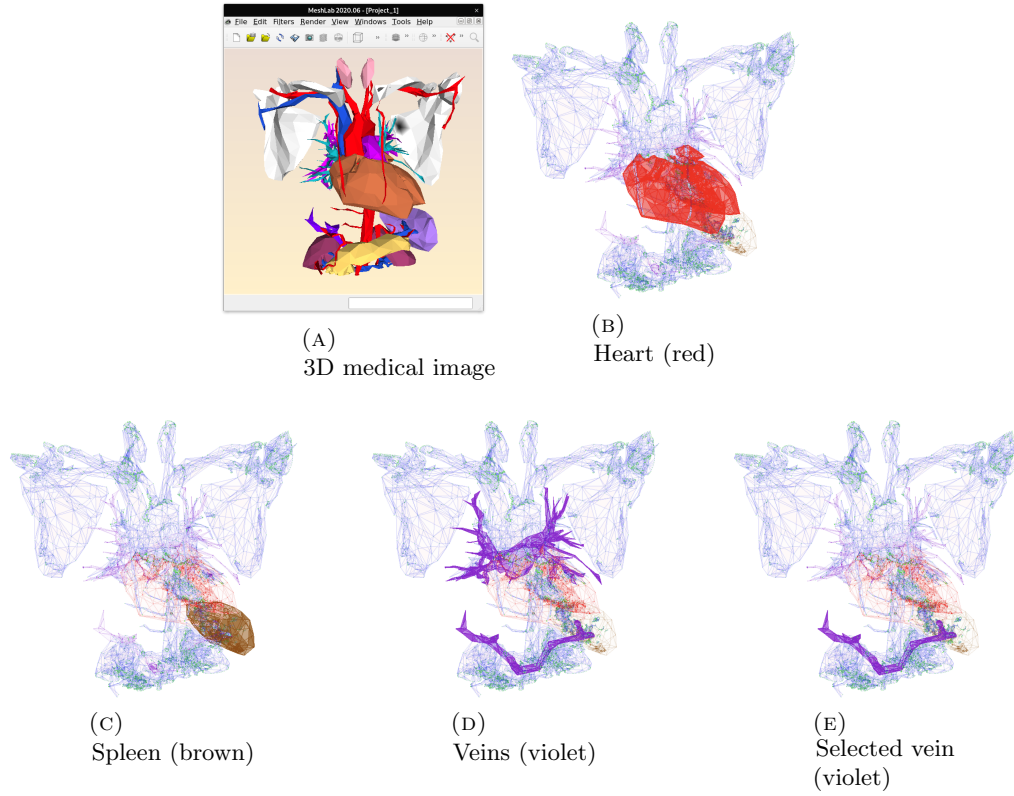
(D)
Veins (violet)

(E)
Selected vein
(violet)

Figure 16: (16a) A 3D medical illustration, courtesy of `www.sketchfab.com` (copyright:
COEUR et vaissaaux by Chair_Digital_Anatomy – The Unesco Chair of digital
anatomy (Paris University) – is licensed under Creative Commons Attribution,
see `https://creativecommons.org/licenses/by/4.0/legalcode`), visualized
using MeshLab [CCC⁺08]. PolyLogicA is used to segment the heart (16b) shown
in red, spleen (16c) shown in brown, and some veins (16d) shown in violet,
and then to segment a specific vein (the one that reaches the spleen) using
a reachability predicate (16e), also shown in violet. In each image, the non
segmented parts are shown as mostly transparent cells.

(2) for each *simplicial* path π_x, with $\pi_x(0) = x$, there is a simplicial path π_y with $\pi_y(0) = y$,
   and $\pi_x \hat{\sim} \pi_y$;
(3) for each *simplicial* path π_y, with $\pi_y(0) = y$, there is a simplicial path π_x with $\pi_x(0) = x$,
   and $\pi_x \hat{\sim} \pi_y$.

The largest simplicial bisimulation, if it exists, is called *simplicial bisimilarity*.

The notion of simplicial bisimulation resembles that of stuttering equivalence for process
calculi [BCG88, DV95, GJKW17] but it should be noted that it is defined for *continuous*
structures. In the following, we state the three main facts that conclude this section. Detailed
proofs can be found in Appendix A.

Theorem 6.3. *Logical equivalence is a simplicial bisimulation.*

Theorem 6.4. *Each simplicial bisimulation is included in logical equivalence.*
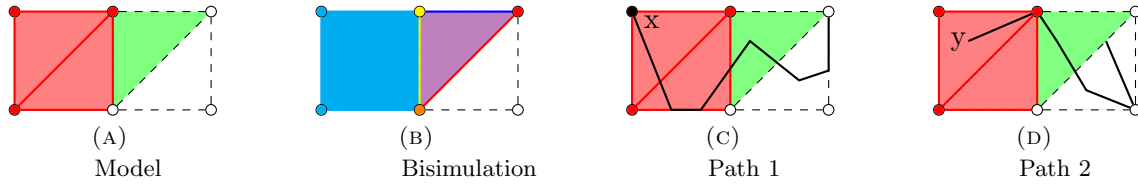
Figure 17: An example of bisimilarity. 17a) Model with atomic propositions in green and red. 17b) bisimilarity, encoded via colours (points of the same colour are bisimilar). 17c) A point x and a simplicial path starting from x. 17d) Another point y, bisimilar to x, and a simplicial path starting from y. The two paths are, in turn, bisimilar; note that these paths are also piecewise linear. The two paths cross a different set of cells, and have a different number of segments.

Corollary 6.5 (of Theorem 6.3 and 6.4). *In a polyhedral model, the largest simplicial bisimulation always exists, and it coincides with logical equivalence.*

Example 6.6. Consider the polyhedral, 1-dimensional model with cells the points $x = -1$, $y = 0$, $z = 1$, and the open segments $s = (-1, 0)$ and $t = (0, 1)$. Consider the set of atomic propositions $\{a, b\}$. Let $V(a) = \{y\}$ and $V(b) = z$. According to topo-bisimilarity [BB07], which characterises the modal fragment of our language, all the points in $s \cup t$ are equivalent, as there is no modal formula telling s and t apart. However, if γ is added to the picture, let $\phi = \gamma(\neg a, b)$. The points of s do not satisfy ϕ, but the points of t do. No point of s is bisimilar to a point of t.

Example 6.7. In Figure 17, we propose a simple illustration of the concept of simplicial bisimilarity. Note how the two presented paths pass through a different number of equivalent cells (in a way akin to classical "stuttering" forms of bisimilarities for process calculi [BK08]).

7. Conclusions and Future Work

We proposed an interpretation of the spatial logic SLCS in continuous space using models based on polyhedra. SLCS is a spatial logic stemming from topological modal logics enhanced with reachability. We developed a polyhedral semantics for SLCS introducing a finite state relational Kripke model for any given polyhedral model. Such Kripke models represent the face-relation between the simplicial complexes of a polyhedron. We showed that such Kripke models provide a full logical invariant for the polyhedral model with respect to SLCS. Based on these theoretical results we presented a novel spatial model checking algorithm for the verification of SLCS properties of polyhedral spaces, and in particular for the reachability operator of SLCS. First applications of the prototype polyhedral model checker PolyLogicA on examples of realistic size showed very encouraging results for the feasibility of this novel type of spatial model checking requiring only about 5 seconds for the actual model checking task of a full-size model consisting of circa 1.5 million of simplicial cells.

Future work will span several theoretical and applied directions. Regarding bisimilarity, by looking at the *simplicial* paths that are used in Definition 6.2, one may wonder what logical operators (alternative to γ), and in what classes of models, can be characterised by lifting the restrictions or completely changing the kind of paths that are used therein. Furthermore, bisimilarity hints at *minimization* in order to reduce the complexity of the analysis. The

preliminary results presented in [CLMV20], including the tool *MiniLogicA* could be useful in this research direction. Note that the quotient mapping each simplex in P to a point in $\widetilde{\mathrm{K}}$ in Definition 4.1 is an *open map*, thus it preserves and reflects logical equivalence of the modal fragment of our language; additionally, as shown in Section 6, it preserves and reflects logical equivalence of the full language, thus simplicial bisimilarity. Not all open maps do so (just consider, e.g., the quotient with respect to classical modal logical equivalence). In future work, we plan to formalise the conditions on an arbitrary open map that make it preserve and reflect simplicial bisimilarity. The relationship between spatial logics and temporal logics, and related bisimilarities [KR97] is also of interest, and in particular, comparing path-based spatial notions such as simplicial bisimilarity, to the so-called *stuttering* equivalences, and their associated minimisation algorithms (see e.g. [BCG88, GJKW17]).

Spatio-temporal model checking in the style of [Gri16, CGL$^+$15] is a planned future development, the simplest case being the one where the underlying polyhedron does not change over time, and only the valuation of atomic propositions depends upon the temporal state of a system. More complex forms of dynamic spatial structures where the underlying polyhedron evolves over time are also of interest.

A promising application of PolyLogicA is fully automated, declarative analysis of 3D *meshes*. Clearly, we foresee 3D medical imaging to be a promising landscape for future applications. Furthermore, note that 3D meshes play a central role in several other fields, including architecture and computer-aided design (CAD), geographic information systems (GIS), see e.g., [BBJ$^+$20], or the entertainment industry (consider 3D games or 3D animation movies), in education and in scientific visualisation.

Implementation-wise, *GPU computing* could provide a computational boost to PolyLogicA. See [BCG21] for a GPU implementation of the parent tool VoxLogicA. Finally, a user interface could be useful to explore large datasets, and to better visualise the interpretation of logic formulas, possibly exploiting results in [BMÖ19] for validation.

References

[ABGM21]   S. Adam-Day, N. Bezhanishvili, D. Gabelaia, and V. Marra. Polyhedral completeness of intermediate logics: the nerve criterion. *CoRR*, abs/2112.07518, 2021. URL: `https://arxiv.org/abs/2112.07518`, `arXiv:2112.07518`.

[AD19]   S. Adam-Day. Polyhedral completeness in intermediate and modal logics. Master's Thesis. Available as ILLC report: MoL-2019-08, 2019. URL: `https://msclogic.illc.uva.nl/theses/archive/publication/4818/Polyhedral-Completeness-in-Intermediate-and-Modal-Logics`.

[AGH$^+$19]   R. Alur, M. Giacobbe, Th. A. Henzinger, K. G. Larsen, and M. Mikucionis. Continuous-time models for system design and analysis. In *Computing and Software Science - State of the Art and Perspectives*, volume 10000 of *Lecture Notes in Computer Science*, pages 452–477. Springer, 2019. `doi:10.1007/978-3-319-91908-9_22`.

[Alu11]   R. Alur. Formal verification of hybrid systems. In *Proceedings of the 11th International Conference on Embedded Software, EMSOFT 2011, part of the Seventh Embedded Systems Week, ESWeek 2011, Taipei, Taiwan, October 9-14, 2011*, pages 273–278. ACM, 2011. `doi:10.1145/2038642.2038685`.

[APB07a]   M. Aiello, I. Pratt-Hartmann, and J. Benthem, van. *Handbook of Spatial Logics*. Springer, 2007. `doi:10.1007/978-1-4020-5587-4`.

[APB07b]   M. Aiello, I. Pratt-Hartmann, and J. Benthem, van. What is spatial logic? In *Handbook of Spatial Logics* [APB07a], pages 1–11. `doi:10.1007/978-1-4020-5587-4_1`.

[BB07]   J. Benthem, van and G. Bezhanishvili. Modal logics of space. In *Handbook of Spatial Logics* [APB07a], pages 217–298. `doi:10.1007/978-1-4020-5587-4_5`.

[BBB+04]   O. T. Balovnev, Th. Bode, M. Breunig, A. B. Cremers, W. Müller, G. Pogodaev, S. S. Shumilov, J. Siebeck, A. Siehl, and A. Thomsen. The story of the GeoToolKit - an object-oriented geodatabase kernel system. *GeoInformatica*, 8(1):5–47, 2004. `doi:10.1023/B:GEIN.0000007723.77851.8f`.

[BBC+20]   F. Banci Buonamici, G. Belmonte, V. Ciancia, D. Latella, and M. Massink. Spatial logics and model checking for medical imaging. *Int. J. Softw. Tools Technol. Transf.*, 22(2):195–217, 2020. `doi:10.1007/s10009-019-00511-9`.

[BBC+21]   G. Belmonte, G. Broccia, V. Ciancia, D. Latella, and M. Massink. Feasibility of spatial model checking for nevus segmentation. In *9th IEEE/ACM International Conference on Formal Methods in Software Engineering, FormaliSE@ICSE 2021, Madrid, Spain, May 17-21, 2021*, pages 1–12. IEEE, 2021. `doi:10.1109/FormaliSE52586.2021.00007`.

[BBJ+20]   M. Breunig, P. E. Bradley, M. Jahn, P. Kuper, N. Mazroob, N. Rösch, M. Al-Doori, E. Stefanakis, and M. Jadidi. Geospatial data management research: Progress and future directions. *ISPRS Int. J. Geo Inf.*, 9(2):95, 2020. `doi:10.3390/ijgi9020095`.

[BCG88]   M. C. Browne, E. M. Clarke, and O. Grumberg. Characterizing finite Kripke structures in propositional temporal logic. *Theor. Comput. Sci.*, 59:115–131, 1988. `doi:10.1016/0304-3975(88)90098-9`.

[BCG21]   L. Bussi, V. Ciancia, and F. Gadducci. Towards a spatial model checker on GPU. In *Formal Techniques for Distributed Objects, Components, and Systems - 41st IFIP WG 6.1 International Conference, FORTE 2021, Held as Part of the 16th International Federated Conference on Distributed Computing Techniques, DisCoTec 2021, Valletta, Malta, June 14-18, 2021, Proceedings*, volume 12719 of *Lecture Notes in Computer Science*, pages 188–196. Springer, 2021. `doi:10.1007/978-3-030-78089-0_12`.

[BCLM19a]   G. Belmonte, V. Ciancia, D. Latella, and M. Massink. Innovating medical image analysis via spatial logics. In *From Software Engineering to Formal Methods and Tools, and Back - Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday*, volume 11865 of *Lecture Notes in Computer Science*, pages 85–109. Springer, 2019. `doi:10.1007/978-3-030-30985-5_7`.

[BCLM19b]   G. Belmonte, V. Ciancia, D. Latella, and M. Massink. VoxLogicA: A spatial model checker for declarative image analysis. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS*, volume 11427 of *Lecture Notes in Computer Science*, pages 281–298. Springer, 2019. `doi:10.1007/978-3-030-17462-0_16`.

[Ben84]   J. Benthem, van. *Correspondence Theory*, pages 167–247. Springer Netherlands, Dordrecht, 1984. `doi:10.1007/978-94-009-6259-0_4`.

[BFGH17]   S. Bogomolov, G. Frehse, M. Giacobbe, and Th. A. Henzinger. Counterexample-guided refinement of template polyhedra. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017*, volume 10205 of *Lecture Notes in Computer Science*, pages 589–606, 2017. `doi:10.1007/978-3-662-54577-5_34`.

[BK08]   C. Baier and J. Katoen. *Principles of model checking*. MIT Press, 2008. URL: `https://mitpress.mit.edu/9780262026499/principles-of-model-checking/`.

[BM14]   J.-D. Boissonnat and C. Maria. The simplex tree: An efficient data structure for general simplicial complexes. *Algorithmica*, 70(3):406–427, Nov 2014. `doi:10.1007/s00453-014-9887-3`.

[BMMP18]   N. Bezhanishvili, V. Marra, D. McNeill, and A. Pedrini. Mckin's theorem on intuitionistic logic, for polyhedra. *Ann. Pure Appl. Log.*, 169(5):373–391, 2018. `doi:10.1016/j.apal.2017.12.005`.

[BMÖ19]   G. Broccia, P. Milazzo, and P. Csaba Ölveczky. Formal modeling and analysis of safety-critical human multitasking. *Innov. Syst. Softw. Eng.*, 15(3-4):169–190, 2019. `doi:10.1007/s11334-019-00333-7`.

[CBLM21]   V. Ciancia, G. Belmonte, D. Latella, and M. Massink. A hands-on introduction to spatial model checking using VoxLogicA. In *Model Checking Software*, pages 22–41, Cham, 2021. Springer International Publishing. `doi:10.1007/978-3-030-84629-9_2`.

[CCC+08]   P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. `doi:10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136`.

[CGG+18]   V. Ciancia, S. Gilmore, G. Grilletti, D. Latella, M. Loreti, and M. Massink. Spatio-temporal model checking of vehicular movement in public transport systems. *Int. J. Softw. Tools Technol. Transf.*, 20(3):289–311, 2018. `doi:10.1007/s10009-018-0483-8`.

[CGL+14]   V. Ciancia, S. Gilmore, D. Latella, M. Loreti, and M. Massink. Data verification for collective adaptive systems: Spatial model-checking of vehicle location data. In *Eighth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops, SASOW*, pages 32–37. IEEE Computer Society, 2014. `doi:10.1109/SASOW.2014.16`.

[CGL+15]   V. Ciancia, G. Grilletti, D. Latella, M. Loreti, and M. Massink. An experimental spatio-temporal model checker. In *Software Engineering and Formal Methods - SEFM 2015 Collocated Workshops*, volume 9509 of *Lecture Notes in Computer Science*, pages 297–311. Springer, 2015. `doi:10.1007/978-3-662-49224-6_24`.

[CLLM14]   V. Ciancia, D. Latella, M. Loreti, and M. Massink. Specifying and verifying properties of space. In *Theoretical Computer Science - 8th IFIP TC 1/WG 2.2 International Conference, TCS 2014, Rome, Italy, September 1-3, 2014. Proceedings*, volume 8705 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2014. `doi:10.1007/978-3-662-44602-7_18`.

[CLLM16a] V. Ciancia, D. Latella, M. Loreti, and M. Massink. Model Checking Spatial Logics for Closure Spaces. *Logical Methods in Computer Science*, Volume 12, Issue 4, October 2016. URL: `http://lmcs.episciences.org/2067`, `doi:10.2168/LMCS-12(4:2)2016`.

[CLLM16b] V. Ciancia, D. Latella, M. Loreti, and M. Massink. Spatial logic and spatial model checking for closure spaces. In *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*, volume 9700 of *Lecture Notes in Computer Science*, pages 156–201. Springer, 2016. `doi:10.1007/978-3-319-34096-8_6`.

[CLM+16]   V. Ciancia, D. Latella, M. Massink, R. Paškauskas, and A. Vandin. A tool-chain for statistical spatio-temporal model checking of bike sharing systems. In *Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques - 7th International Symposium, ISoLA 2016, Part I*, volume 9952 of *Lecture Notes in Computer Science*, pages 657–673, 2016. `doi:10.1007/978-3-319-47166-2_46`.

[CLMP15]   V. Ciancia, D. Latella, M. Massink, and R. Paškauskas. 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. `doi:10.1109/SASOW.2015.17`.

[CLMV20]   V. Ciancia, D. Latella, M. Massink, and E. P. Vink, de. Towards spatial bisimilarity for closure models: Logical and coalgebraic characterisations. *CoRR*, abs/2005.05578, 2020. URL: `https://arxiv.org/abs/2005.05578`, `arXiv:2005.05578`.

[CM21]   D. Castelnovo and M. Miculan. Closure hyperdoctrines. In *9th Conference on Algebra and Coalgebra in Computer Science, CALCO 2021, August 31 to September 3, 2021, Salzburg, Austria*, volume 211 of *LIPIcs*, pages 12:1–12:21. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. `doi:10.4230/LIPIcs.CALCO.2021.12`.

[DV95]   R. De Nicola and F. W. Vaandrager. Three logics for branching bisimulation. *J. ACM*, 42(2):458–487, 1995. `doi:10.1145/201019.201032`.

[Gal99]   A. Galton. The mereotopology of discrete space. In *Spatial Information Theory. Cognitive and Computational Foundations of Geographic Information Science*, volume 1661 of *Lecture Notes in Computer Science*, pages 251–266. Springer, 1999. URL: `http://dx.doi.org/10.1007/3-540-48384-5_17`, `doi:10.1007/3-540-48384-5_17`.

[Gal14]   A. Galton. Discrete mereotopology. In *Mereology and the Sciences: Parts and Wholes in the Contemporary Scientific Context*, pages 293–321. Springer International Publishing, 2014. `doi:10.1007/978-3-319-05356-1_11`.

[GJKW17]   J. F. Groote, D. N. Jansen, J. J. A. Keiren, and A. Wijs. An $O(m\log n)$ algorithm for computing stuttering equivalence and branching bisimulation. *ACM Trans. Comput. Log.*, 18(2):13:1–13:34, 2017. `doi:10.1145/3060140`.

[GL08]   A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In *Hybrid Systems: Computation and Control, 11th International Workshop, 2008.*, volume 4981 of *Lecture Notes in Computer Science*, pages 215–228. Springer, 2008. `doi:10.1007/978-3-540-78929-1_16`.

[Gri16]   G. Grilletti. Spatio-temporal model checking: Explicit and abstraction-based methods. Master's thesis, University of Pisa, 2016. URL: `https://etd.adm.unipi.it/t/etd-06282016-191103/`.

[GSC+09] R. Grosu, S. Smolka, F. Corradini, A. Wasilewska, E. Entcheva, and E. Bartocci. Learning and detecting emergent behavior in networks of cardiac myocytes. *Commun. ACM*, 52(3):97–105, 2009. `doi:10.1145/1467247.1467271`.

[Hen00]   Th. A. Henzinger. The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, pages 265–292. Springer, 2000. `doi:10.1007/978-3-642-59615-5_13`.

[HH94]   Th. A. Henzinger and Pei-Hsin Ho. HYTECH: the Cornell HYbrid TECHnology Tool. In *Hybrid Systems II, Proceedings of the Third International Workshop on Hybrid Systems*, volume 999 of *Lecture Notes in Computer Science*, pages 265–293. Springer, 1994. `doi:10.1007/3-540-60472-3_14`.

[HJK+15] I. Haghighi, A. Jones, Z. Kong, E. Bartocci, R. Grosu, and C. Belta. SpaTeL: A novel spatial-temporal logic and its applications to networked systems. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, HSCC '15, pages 189–198. ACM, 2015. `doi:10.1145/2728606.2728633`.

[KR97]   N. Kurtonina and M. Rijke, de. Bisimulations for temporal logic. *J. Log. Lang. Inf.*, 6(4):403–425, 1997. `doi:10.1023/A:1008223921944`.

[LPS20]   S. Linker, F. Papacchini, and M. Sevegnani. Analysing spatial properties on neighbourhood spaces. In *45th International Symposium on Mathematical Foundations of Computer Science, MFCS 2020, August 24-28, 2020, Prague, Czech Republic*, volume 170 of *LIPIcs*, pages 66:1–66:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. `doi:10.4230/LIPIcs.MFCS.2020.66`.

[LPZ12]   J. A. Levine, R. R. Paulsen, and Y. Zhang. Mesh processing in medical-image analysis – a tutorial. *IEEE Computer Graphics and Applications*, 32(5):22–28, 2012. `doi:10.1109/MCG.2012.91`.

[LQ21]   M. Loreti and M. Quadrini. A spatial logic for a simplicial complex model. *CoRR*, abs/2105.08708, 2021. URL: `https://arxiv.org/abs/2105.08708`, `arXiv:2105.08708`.

[Mau80]   C. R. F. Maunder. Algebraic topology. Cambridge University Press., 1980. URL: `https://www.cambridge.org/it/academic/subjects/mathematics/geometry-and-topology/algebraic-topology-1`.

[MBL+21] M. Ma, E. Bartocci, E. Lifland, J. A. Stankovic, and L. Feng. A novel spatial–temporal specification-based monitoring system for smart cities. *IEEE Internet of Things Journal*, 8(15):11793–11806, 2021. `doi:10.1109/JIOT.2021.3069943`.

[MC21]   A. Muntoni and P. Cignoni. PyMeshLab. *Zenodo*, January 2021. `doi:10.5281/zenodo.4438750`.

[MT44]   J. McKinsey and A. Tarski. The algebra of topology. *Annals of Mathematics*, 45:141–191, 1944. `doi:10.2307/1969080`.

[NBBL22] L. Nenzi, E. Bartocci, L. Bortolussi, and M. Loreti. A Logic for Monitoring Dynamic Networks of Spatially-distributed Cyber-Physical Systems. *Logical Methods in Computer Science*, Volume 18, Issue 1, January 2022. URL: `https://lmcs.episciences.org/8936`, `doi:10.46298/lmcs-18(1:4)2022`.

[NBC+18] L. Nenzi, L. Bortolussi, V. Ciancia, M. Loreti, and M. Massink. Qualitative and Quantitative Monitoring of Spatio-Temporal Properties with SSTL. *Logical Methods in Computer Science*, 14(4):1–38, 2018. DOI 10.23638/LMCS-14(4:2)2018. Published on line: 23 Oct. 2018. ISSN: 1860-5974.

[SDI08]   S. Sankaranarayanan, T. Dang, and F. Ivancic. Symbolic model checking of hybrid systems using template polyhedra. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008.*, volume 4963 of *Lecture Notes in Computer Science*, pages 188–202. Springer, 2008. `doi:10.1007/978-3-540-78800-3_14`.

[TKG17]   C. Tsigkanos, T. Kehrer, and C. Ghezzi. Modeling and verification of evolving cyber-physical spaces. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2017*, pages 38–48. ACM, 2017. `doi:10.1145/3106237.3106299`.

Appendix A. Additional Lemmas and Proofs

Proposition A.1. *Consider the operator ρ of* [BCLM19b]. *We have* $\mathcal{X}, x \vDash \rho\,\psi[\phi] \iff \mathcal{X}, x \models \psi \vee \gamma(\phi, \psi)$, *and* $\mathcal{X}, x \vDash \gamma(\phi, \psi) \iff \mathcal{X}, x \vDash \rho\,(\phi \wedge \rho\,\psi[\phi])[\phi]$.

Proof. We note in passing that the following proof (and the definition of ρ and γ) generalise to arbitrary topological models. We first recall the formal definition of ρ, which uses right-open paths (that is, total continuous functions having $\mathbb{R}_{\geq 0}$ as a domain). Note that in [BCLM19b], ρ was defined only on discrete spaces. However, following the direction of [CLLM16a], the definition applies in a natural way to continuous spaces as follows: $\mathcal{X}, x \vDash \rho\,\psi\,[\phi]$ whenever there is a right-open path $\pi : \mathbb{R}_{\geq 0} \to P$ and an index r such that $\pi(0) = x$, $\pi(r) \in \llbracket \psi \rrbracket^{\mathcal{X}}$ and $\pi((0, r)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$. We proceed by proving the four implications in the statement separately.

First, we show that $\mathcal{X}, x \models \rho\,\psi[\phi]$ implies $\mathcal{X}, x \models \psi \vee \gamma(\phi, \psi)$. If $\mathcal{X}, x \models \rho\,\psi[\phi]$ then there is right-open path π and $\ell \in \mathbb{R}_{\geq 0}$ such that $\pi(0) = x, \pi((0, \ell)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi(\ell) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. We consider two distinct cases:

Case 1: $\ell = 0$. In this case $x \in \llbracket \psi \rrbracket^{\mathcal{X}}$ and so $\mathcal{X}, x \models \psi \vee \gamma(\phi, \psi)$.

Case 2: $\ell \neq 0$. In this case, let π' with $\pi'(r) = \pi(r\ell)$ for all $r \in [0, 1]$. We have $\pi'(0) = \pi(0) = x$, $\pi'((0,1)) = \pi((0,\ell)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi'(1) = \pi(\ell) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. This means that $\mathcal{X}, x \models \gamma(\phi, \psi)$, and so $\mathcal{X}, x \models \psi \vee \gamma(\phi, \psi)$.

We now show that $\mathcal{X}, x \models \psi \vee \gamma(\phi, \psi)$ implies $\mathcal{X}, x \models \rho\,\psi[\phi]$. Suppose that $\mathcal{X}, x \models \psi \vee \gamma(\phi, \psi)$. We consider two distinct cases:

Case 1: $\mathcal{X}, x \models \psi$. In this case it trivially holds $\mathcal{X}, x \models \rho\,\psi[\phi]$ (for any right-open path starting at x, just consider $\ell = 0$).

Case 2: $\mathcal{X}, x \models \gamma(\phi, \psi)$. In this case there is a path π such that $\pi(0) = x$, $\pi((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. We obtain $\mathcal{X}, x \models \rho\,\psi[\phi]$ by taking $\ell = 1$ in the definition of the semantic clause of ρ.

We now show that $\mathcal{X}, x \models \gamma(\phi, \psi)$ implies $\mathcal{X}, x \models \rho\,(\phi \wedge \rho\,\psi[\phi])[\phi]$. Suppose $\mathcal{X}, x \models \gamma(\phi, \psi)$, that is, there is a path π such that $\pi(0) = x$, $\pi((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. Since π is a total continuous function, there is an $\epsilon \in (0, 1)$ such that $\pi(\epsilon) \in \llbracket \phi \rrbracket^{\mathcal{X}}$. We define two additional paths $\pi_1(t) = \pi(t\epsilon)$ and $\pi_2(t) = \pi((1 - \epsilon)t + \epsilon)$. Notice that $\pi_2(0) = \pi(\epsilon) \in \llbracket \phi \rrbracket^{\mathcal{X}}, \pi_2((0,1)) = \pi((\epsilon, 1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi_2(1) = \pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$: this shows that $\mathcal{X}, \pi(\epsilon) \models \phi \wedge \rho\,\psi[\phi]$. Moreover, we have $\pi_1(0) = \pi(0) = x, \pi_1((0,1)) = \pi((0,\epsilon)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi_1(1) = \pi_2(0) \in \llbracket \phi \wedge \rho\,\psi[\phi] \rrbracket^{\mathcal{X}}$: this shows that $\mathcal{X}, x \models \rho\,(\phi \wedge \rho\,\psi[\phi])[\phi]$.

Finally, we show that $\mathcal{X}, x \models \rho\,(\phi \wedge \rho\,\psi[\phi])[\phi]$ implies $\mathcal{X}, x \models \gamma(\phi, \psi)$. Suppose that $\mathcal{X}, x \models \rho\,(\phi \wedge \rho\,\psi[\phi])[\phi]$, that is, there is a right-open path π_1 and a value $\ell_1 \in \mathbb{R}_{\geq 0}$ such that $\pi_1(0) = x, \pi_1((0, \ell_1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi_1(\ell_1) \in \llbracket \phi \wedge \rho\,\psi[\phi] \rrbracket^{\mathcal{X}} = \llbracket \phi \rrbracket^{\mathcal{X}} \cap \llbracket \rho\,\psi[\phi] \rrbracket^{\mathcal{X}}$. By the last condition, there is a right-open path π_2 and a value $\ell_2 \in \mathbb{R}_{\geq 0}$ such that $\pi_2(0) = \pi_1(\ell_1)$, $\pi_2((0, \ell_2)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi_2(\ell_2) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. We consider two distinct cases:

Case 1: $l_1 = l_2 = 0$. In this case $\pi_1(l_1) = \pi_2(l_2) = x$, and so $x \in \llbracket \phi \rrbracket^{\mathcal{X}} \cap \llbracket \psi \rrbracket^{\mathcal{X}}$. In this case $\mathcal{X}, x \vDash \gamma(\phi, \psi)$ holds trivially.

Case 2: $l_1 > 0$ or $l_2 > 0$. In this case define the path π' by imposing $\pi'(t) = \pi_1(2tl_1)$ and $\pi'(\frac{1}{2} + t) = \pi_2(2tl_2)$ for $t \in [0, \frac{1}{2}]$ (notice that the path is well-defined since $\pi'(\frac{1}{2}) = \pi_1(l_1) = \pi_2(0)$). Clearly $\pi'(0) = x$. Moreover, $\pi'((0,1)) = \pi'((0, \frac{1}{2}]) \cup \pi'((\frac{1}{2}, 1)) = \pi_1((0, l_1)) \cup \pi_2((0, l_2)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ (under the assumption that $l_1 > 0$ or $l_2 > 0$). And finally $\pi'(1) = \pi_2(l_2) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. This shows that $\mathcal{X}, x \vDash \gamma(\phi, \psi)$, as desired. $\square$

Proposition A.2. *Let σ be a non-empty simplex and consider $x \in \widetilde{\sigma}$ and $y \in \sigma$. Then there is a* linear *path (a segment in $\mathbb{R}^d$) $\pi : [0,1] \to \sigma$ with $\pi(0) = x$, $\pi(1) = y$ and $\pi([0,1)) \subseteq \widetilde{\sigma}$; more precisely, $\pi(t) = ty + (1-t)x$.*

Proof. Let $V = \{v_0, \ldots, v_n\}$ be the set of vertices of σ. By definition of σ, x and y are in the convex hull of V, that is, there exist λ_i and δ_i such that $x = \sum_{i=0}^{n} \lambda_i v_i$ and $y = \sum_{i=0}^{n} \delta_i v_i$. Since $x \in \widetilde{\sigma}$, every λ_i is strictly greater than 0. For π as defined in the statement we have $\pi(t) = \sum_{i=0}^{n} (t\delta_i + (1-t)\lambda_i) \, v_i$. This function is clearly continuous (thus a path), and we have $\pi(0) = x$ and $\pi(1) = y$. Moreover, for every $t \in [0,1)$ and every $i \leq n$ we have $t\delta_i + (1-t)\lambda_i \geq (1-t)\lambda_i > 0$, and so $\pi([0,1)) \subseteq \widetilde{\sigma}$. $\qquad\square$

Lemma A.3. *Let K be a simplicial complex and $\sigma, \tau \in K$. Then $\sigma \preceq \tau$ iff $\widetilde{\sigma} \,\widetilde{\preceq}\, \widetilde{\tau}$ iff $\widetilde{\sigma} \subseteq \mathcal{C}(\widetilde{\tau})$ iff $\widetilde{\sigma} \cap \mathcal{C}(\widetilde{\tau}) \neq \emptyset$.*

Proof. The first equivalence is simply the definition of $\widetilde{\preceq}$ (see Definition 4.2). As for the other equivalences, firstly we show that $\sigma \preceq \tau$ implies $\widetilde{\sigma} \subseteq \mathcal{C}(\widetilde{\tau})$. In fact, if $\sigma \preceq \tau$ we have $\sigma \subseteq \tau$, and so $\widetilde{\sigma} \subseteq \sigma \subseteq \tau = \mathcal{C}(\widetilde{\tau})$. Secondly, note that since cells are nonempty sets we have that $\widetilde{\sigma} \subseteq \mathcal{C}(\widetilde{\tau})$ implies $\widetilde{\sigma} \cap \mathcal{C}(\widetilde{\tau}) \neq \emptyset$. Finally, we show that $\widetilde{\sigma} \cap \mathcal{C}(\widetilde{\tau}) \neq \emptyset$ implies $\sigma \preceq \tau$, concluding the proof. Notice that $\mathcal{C}(\widetilde{\tau}) = \tau = \bigcup \{\widetilde{\tau'} \,|\, \tau' \preceq \tau\}$. So by Lemma 2.4 we have that either $\widetilde{\sigma} \in \{\widetilde{\tau'} \,|\, \tau' \preceq \tau\}$ or $\widetilde{\sigma} \cap \mathcal{C}(\widetilde{\tau}) = \emptyset$. Since the latter is not the case by assumption, we conclude $\sigma \preceq \tau$. $\qquad\square$

Lemma A.4. *Let π_1 and π_2 be two paths, with $\pi_1(1) = \pi_2(0)$; let $x, y \in [0,1]$, with $x < y$. Define in the obvious way the* concatenation *$\pi_1 ; \pi_2$, with $\pi_1 ; \pi_2(0) = \pi_1(0)$ and $\pi_1 ; \pi_2(1) = \pi_2(1)$ and the sub-path $\pi_1^{[x,y]}$, with $\pi_1^{[x,y]}(0) = \pi_1(x)$ and $\pi_1^{[x,y]}(1) = \pi_1(y)$. We have that: whenever π_1 and π_2 are piecewise-linear (simplicial), also $\pi_1 ; \pi_2$ is piecewise linear (simplicial); whenever π_1 is piecewise linear (simplicial), also $\pi_1^{[x,y]}$ is piecewise linear (simplicial). Furthermore, for any two (additional) paths π_1', π_2' with $\pi_1'(1) = \pi_2'(0)$, and relation R, if $\pi_1 \hat{R} \pi_1'$ and $\pi_2 \hat{R} \pi_2'$, then $\pi_1 ; \pi_2 \hat{R} \pi_1' ; \pi_2'$.*

Proof. We omit the proof, which is straightforward. $\qquad\square$

Lemma 3.4. *Let $\mathcal{X} = \langle P, K, V \rangle$ and $\mathcal{X}' = \langle P, K', V \rangle$ be two models sharing the same P and V. For each $x \in P$ and ϕ we have: $\mathcal{X}, x \vDash \phi \iff \mathcal{X}', x \vDash \phi$.*

Proof. Just note that K does not appear in Definition 3.3. $\qquad\square$

Lemma 3.6. *Let $\mathcal{X}$ be a polyhedral model and K a simplicial complex coherent with $\mathcal{X}$. Then for each cell $\widetilde{\sigma} \in \widetilde{K}$ and $x, y \in \widetilde{\sigma}$ we have $x \equiv y$.*

Proof. We prove the result by induction on the structure of ϕ. Since most of the cases follow easily from the semantic clauses of the logic, we show only the cases for $\phi = \Box \psi$ and $\phi = \gamma(\psi, \chi)$.

Case $\phi = \Box \psi$: By inductive hypothesis, $[\![\psi]\!]^{\mathcal{X}}$ is a union of cells of $\widetilde{K}$, and by Lemma 2.4 also $P \setminus [\![\psi]\!]^{\mathcal{X}}$ is a union of cells. Since there are only finitely many cells in $\widetilde{K}$, we have $\mathcal{C}(P \setminus [\![\psi]\!]^{\mathcal{X}}) = \bigcup \{\mathcal{C}(\widetilde{\sigma}) \,|\, \widetilde{\sigma} \subseteq P \setminus [\![\psi]\!]^{\mathcal{X}}\}$. And since $\mathcal{C}(\widetilde{\sigma}) = \sigma = \bigcup \{\widetilde{\tau} \,|\, \tau \preceq \sigma\}$, it follows that $\mathcal{C}(P \setminus [\![\psi]\!]^{\mathcal{X}}) = \bigcup \{\widetilde{\tau} \,|\, \exists \sigma. \tau \preceq \sigma \text{ and } \widetilde{\sigma} \subseteq P \setminus [\![\psi]\!]^{\mathcal{X}}\}$, that is, $\mathcal{C}(P \setminus [\![\psi]\!]^{\mathcal{X}})$ is a union of cells. To conclude, by Lemma 2.4 we have that $[\![\phi]\!]^{\mathcal{X}} = \mathcal{I}([\![\psi]\!]^{\mathcal{X}}) = P \setminus \mathcal{C}(P \setminus [\![\psi]\!]^{\mathcal{X}})$ is again a union of cells.

Case $\phi = \gamma(\psi, \chi)$: Suppose that $\mathcal{X}, x \vDash \gamma(\phi, \psi)$: we aim to show that $\mathcal{X}, y \vDash \gamma(\phi, \psi)$. The fact that $\mathcal{X}, x \vDash \gamma(\phi, \psi)$ means that there exists a path π such that $\pi(0) = x$, $\pi((0,1)) \subseteq [\![\phi]\!]^{\mathcal{X}}$

and $\pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. By inductive hypothesis $\llbracket \phi \rrbracket^{\mathcal{X}}$ is a union of cells, and since $\llbracket \phi \rrbracket^{\mathcal{X}} \cap \pi((0,1))$ is not empty also the set $\Pi = \{\widetilde{\tau} | \exists r \in (0,1).\pi(r) \in \widetilde{\tau} \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}\}$ is not empty.

Since $\pi((0,1)) \subseteq \bigcup \Pi$, we have $x \in \mathcal{C}(\pi((0,1))) \subseteq \bigcup\{\tau | \exists r \in (0,1).\pi(r) \in \widetilde{\tau} \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}\}$. Thus there exists a value $r \in (0,1)$ and a cell $\widetilde{\tau}$ such that $x \in \tau$ and $\pi(r) \in \widetilde{\tau}$. Since $x \in \widetilde{\sigma} \cap \mathcal{C}(\widetilde{\tau})$, by Lemma A.3 we have $\widetilde{\sigma} \subseteq \mathcal{C}(\widetilde{\tau}) = \tau$ and consequently $y \in \tau$. By Proposition A.2 (modulo inverting and reparametrising the path) there exists a path $\pi' : [0,r] \to \tau$ such that $\pi'(0) = y$, $\pi'(r) = \pi(r)$ and $\pi'((0,r)) \subseteq \widetilde{\tau} \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$. If we extend π' by imposing $\pi'(t) = \pi(t)$ for $t \in (r,1]$, we obtain a path such that $\pi'(0) = y$, $\pi'((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi'(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. In particular π' witnesses that $\mathcal{X}, y \vDash \gamma(\phi, \psi)$, as desired. □

Proposition 3.7. *Given a polyhedral model $\mathcal{X}$, the relation $\equiv$ has only finitely many equivalence classes. Furthermore, each equivalence class C has a characteristic formula ϕ^C such that $\mathcal{X}, x \vDash \phi^C \iff x \in C$.*

Proof. Fix K coherent with $\mathcal{X}$. By Lemma 3.6, each equivalence class is a union of distinct cells of $\widetilde{K}$. But since there are only finitely many cells, there are finitely many equivalence classes too. For two distinct equivalence classes C and C', let $\phi^{C,C'}$ be a formula satisfied by the elements of C but not by the elements in C'—such a formula exists by definition of $\equiv$. Then the formula $\phi^C = \bigwedge_{C' \neq C} \phi^{C,C'}$ is satisfied only by elements in C, as desired. □

Theorem 3.8. *For each formula ϕ, we have $\mathcal{X}, x \vDash \Box \phi \iff \mathcal{X}, x \vDash \neg\gamma(\neg\phi, \top)$.*

Proof. Recall the definition of $\Diamond\phi := \neg\Box\neg\phi$ and that $\mathcal{X}, x \models \Diamond\phi \iff x \in \mathcal{C}(\llbracket \phi \rrbracket^{\mathcal{X}})$ (see e.g. [BB07]), where $\mathcal{C}$ is the topological closure operator. Thus, we prove the equivalent statement $\mathcal{X}, x \vDash \Diamond\phi \iff \mathcal{X}, x \vDash \gamma(\phi, \top)$.

If $\mathcal{X}, x \models \gamma(\phi, \top)$, by definition, there is a path π with $\pi(0) = x$ and $\pi((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$. In particular we have $x = \pi(0) \in \mathcal{C}(\pi((0,1)))$ by continuity of paths, and $\mathcal{C}(\pi((0,1))) \subseteq \mathcal{C}(\llbracket \phi \rrbracket^{\mathcal{X}})$ by monotonicity of $\mathcal{C}$. Therefore $x \in \mathcal{C}(\llbracket \phi \rrbracket^{\mathcal{X}})$, which amounts to $\mathcal{X}, x \vDash \Diamond\phi$.

If $\mathcal{X}, x \models \Diamond\phi$, we have $x \in \mathcal{C}(\llbracket \phi \rrbracket^{\mathcal{X}})$. Fix K a simplicial complex coherent with $\mathcal{X}$. By Lemma 3.6 $\llbracket \phi \rrbracket^{\mathcal{X}}$ is a union of cells in $\widetilde{K}$, and so $\mathcal{C}(\llbracket \phi \rrbracket^{\mathcal{X}}) = \bigcup\{\mathcal{C}(\widetilde{\sigma}) | \widetilde{\sigma} \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}\}$. In particular, $x \in \mathcal{C}(\widetilde{\sigma}) = \sigma$ for one of these cells. Fix an arbitrary element $y \in \widetilde{\sigma}$—recall that $\widetilde{\sigma}$ is nonempty whenever σ is nonempty. By Proposition A.2 (note that the names x and y are inverted in the statement of the proposition) there is a path π with $\pi(0) = y$, $\pi(1) = x$, and $\pi([0,1)) \subseteq \widetilde{\sigma} \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$. The "converse" path $\pi'(i) := \pi(1-i)$ witnesses that $\mathcal{X}, x \models \gamma(\phi, \top)$. □

Lemma 3.10. *Let K be a simplicial complex and $x, y \in |K|$. Then there exists a path in $|K|$ from x to y if and only if there is a PL-path in $|K|$ from x to y.*

Proof. The right-to-left direction is trivial, so we focus on the left-to-right direction. Fix a path in $|K|$ starting at x and ending at y, and consider the set $\Pi \subseteq \widetilde{K}$ of cells traversed by the path. Clearly $\bigcup \Pi$ is a connected subset of $|K|$. To obtain the desired result it suffices to show that $\bigcup \Pi$ is also *PL-connected*, that is, for all $x, y \in \bigcup \Pi$ there exists a PL-path from x to y. We are going to prove this by induction on the cardinality of Π.

Base case: If $\#\Pi = 1$, then any two points $x, y \in \bigcup \Pi$ belong to the same cell $\widetilde{\sigma}$, and since cells are convex there is a linear path (thus piecewise linear) connecting x and y.

Inductive step: Suppose that $\#\Pi = n + 1$ and that the result holds for sets with lower cardinality. Consider two points $x, z \in \bigcup \Pi$ and call $\widetilde{\sigma}$ the cell containing x. If $z \in \widetilde{\sigma}$ too we

can reason as in the base case, so we can assume otherwise. Define Z to be the connected component of $\bigcup(\Pi \setminus \{\widetilde{\sigma}\})$ containing z. Z is itself the union of a set of cells Π'—since cells are connected. As $\#\Pi' < \#\Pi$, by inductive hypothesis Z is PL-connected. Notice that $Y := \widetilde{\sigma} \cup Z$ is connected, for otherwise Z would be disconnected from $\bigcup\Pi \setminus Z$, against the initial assumption that $\bigcup\Pi$ is connected.

$\mathcal{C}(\widetilde{\sigma}) = \sigma$ and $\mathcal{C}(Z)$ are closed sets whose union covers Y. If the intersection $Y \cap \sigma \cap \mathcal{C}(Z)$ were empty, then $Y \cap \sigma$ and $Y \cap \mathcal{C}(Z)$ would disconnect Y. So there must be a point $y \in Y \cap \sigma \cap \mathcal{C}(Z)$. By Proposition A.2 there exists a linear (thus PL) path π_1 connecting $x \in \widetilde{\sigma}$ and $y \in \sigma$, and with $\pi_1([0,1)) \subseteq \widetilde{\sigma} \subseteq Y$. Moreover, since $y \in \mathcal{C}(Z) = \mathcal{C}(\bigcup\Pi')$, there exists a cell $\widetilde{\tau} \in \Pi'$ such that $y \in \mathcal{C}(\widetilde{\tau}) = \tau$. So, again by Proposition A.2 (with w playing the role of x in the statement of the proposition), there exists a linear path π_2 from $y \in \tau$ to an arbitrary point $w \in \widetilde{\tau}$, fully contained in Y. Finally, since Z is PL-connected, there exists a PL-path π_3 from w to z in $Z \subseteq Y$. By concatenating π_1, π_2 and π_3 we obtain a PL-path connecting x and z contained in $Y \subseteq \bigcup\Pi$, as desired. $\square$

Lemma 3.11. *We have:* $\mathcal{X}, x \vDash \gamma(\phi, \psi)$ *if and only if there is a PL-path* $\pi : [0,1] \overset{PL}{\to} P$ *such that* $\pi(0) = x$ *and* $\pi((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ *and* $\pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$.

Proof. The right-to-left implication follows trivially from the semantics clauses of the reachability operators, so we focus on the left-to-right implications. Fix a simplicial complex K coherent with $\mathcal{X}$. Suppose that there exists a path $\pi : [0,1] \to P$ such that $\pi(0) = x$, $\pi((0,1)) \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$. Consider the set of cells $\Pi := \{\widetilde{\sigma} \in \widetilde{K} \mid \pi((0,1)) \cap \widetilde{\sigma} \neq \emptyset\}$. Clearly $\bigcup\Pi$ is connected, and so by Lemma 3.10 it is also PL-connected (i.e., for every two points in $\bigcup\Pi$ there is a PL-path in $\bigcup\Pi$ connecting them). Moreover $x, \pi(1) \in \mathcal{C}(\bigcup\Pi)$.

Since $x \in \mathcal{C}(\bigcup\Pi)$, there exists a cell $\widetilde{\sigma} \in \Pi$ such that $x \in \mathcal{C}(\widetilde{\sigma}) = \sigma$. So by Proposition A.2 there exists a linear path π_1 from x to a point (arbitrarily chosen) $y \in \widetilde{\sigma} \subseteq \bigcup\Pi$ such that $\pi_1((0,1)) \subseteq \widetilde{\sigma}$ (note that the names x and y are inverted in the statement of the proposition). By a similar argument, there exists a linear path π_3 from a point $z \in \bigcup\Pi$ to $\pi(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$, such that $\pi_3((0,1)) \subseteq \bigcup\Pi$. Finally, since $\bigcup\Pi$ is PL-connected, there exists a PL-path π_2 from y to z completely contained in $\bigcup\Pi$. By concatenating π_1, π_2 and π_3 we obtain a PL-path π' such that $\pi'(0) = x$, $\pi'((0,1)) \subseteq \bigcup\Pi \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$ and $\pi'(1) \in \llbracket \psi \rrbracket^{\mathcal{X}}$, as desired. $\square$

Lemma 3.13. *Any piecewise linear path is simplicial.*

Proof. As cells are convex sets, the intersection between a cell and a segment is a segment. So any segment crosses each cell at most once. Observing that there are finitely many segments in a PL-path, one obtains the proof. $\square$

Theorem 4.4. *Let* $\mathcal{X} = \langle P, K, V \rangle$ *be a polyhedral model and x a point of P. Let $\sigma \in K$ be the unique simplex such that $x \in \widetilde{\sigma}$. For every formula ϕ of SLCS we have* $\mathcal{X}, x \vDash \phi \iff \mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \phi$.

Proof. Notice that by Lemma 3.6 the left side of the bi-implication is equivalent to $\widetilde{\sigma} \subseteq \llbracket \phi \rrbracket^{\mathcal{X}}$. We proceed by induction on the structure of the formula; the only non trivial cases are when the formula is of the form $\Box\psi$ and when the formula is of the form $\gamma(\phi, \psi)$.

Case $\Box\psi$: We have that $\mathcal{X}, x \vDash \Box\psi$ iff $x \notin \mathcal{C}(\llbracket \neg\psi \rrbracket^{\mathcal{X}})$. By Lemma 3.6, $\llbracket \neg\psi \rrbracket^{\mathcal{X}}$ is a finite union of cells, so, by Lemma A.3, the condition is equivalent to $x \notin \bigcup\{\tau \mid \widetilde{\tau} \subseteq \llbracket \neg\psi \rrbracket^{\mathcal{X}}\}$; and again, by Lemma A.3, it is equivalent to $\widetilde{\sigma} \npreceq \widetilde{\tau}$ for every $\widetilde{\tau} \subseteq \llbracket \neg\psi \rrbracket^{\mathcal{X}}$. Again by Lemma 3.6, for every $\widetilde{\tau} \in \widetilde{K}$ we have that if $\widetilde{\sigma} \preceq \widetilde{\tau}$ then $\widetilde{\tau} \subseteq \llbracket \psi \rrbracket^{\mathcal{X}}$, which by the inductive hypothesis

amounts to $\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \Box\psi$. Note that we have equivalences in both directions of the proof, so the bi-implication is obtained "for free" for this part.

Case $\gamma(\phi, \psi)$**:** Firstly, suppose that $\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \gamma(\phi, \psi)$. Then there exists a $\pm$-path $\pi : \{0, \ldots, k\} \overset{\pm}{\to} \widetilde{\mathrm{K}}$ such that $\pi(0) = \widetilde{\sigma}$, $\pi(\{1, \ldots, k-1\}) \subseteq \llbracket\phi\rrbracket^{\mathcal{M}}$ and $\pi(k) \in \llbracket\psi\rrbracket^{\mathcal{M}}$.

By inductive hypothesis and Lemma 3.6, $\pi(j) \subseteq \llbracket\phi\rrbracket^{\mathcal{X}}$ for $j \in \{1, \ldots, k-1\}$ and $\pi(k) \subseteq \llbracket\psi\rrbracket^{\mathcal{X}}$.

For any two cells $\widetilde{\sigma} \overset{\sim}{\preceq} \widetilde{\tau}$, we have $b_\sigma \in \widetilde{\sigma} \subseteq \tau$ (recall that b_σ indicates the *barycentre* of σ). So by Proposition A.2 (where b_σ and b_τ play the role of y and x respectively in the statement of the proposition) there exists a linear path connecting $b_\sigma \in \tau$ to $b_\tau \in \widetilde{\tau}$ and mapping the interval $(0, 1)$ to $\widetilde{\tau}$. Using this fact, for every $i = 1, \ldots k$ we can find a linear path π_i' connecting $b_{\pi(i-1)}$ to $b_{\pi(i)}$ with $\pi_i'((0, 1)) \subseteq \pi(i-1)$ or $\pi_i'((0, 1)) \subseteq \pi(i)$—depending on whether $\pi(i-1) \overset{\sim}{\preceq} \pi(i)$ or $\pi(i) \overset{\sim}{\preceq} \pi(i-1)$. Concatenating these paths, we obtain a path $\pi' := \pi_1'; \ldots; \pi_k'$ from $b_{\pi(0)}$ to $b_{\pi(k)} \in \llbracket\psi\rrbracket^{\mathcal{X}}$ such that $\pi'((0, 1)) \subseteq \pi(1) \cup \cdots \cup \pi(k-1) \subseteq \llbracket\phi\rrbracket^{\mathcal{X}}$. This shows that $\mathcal{X}, b_{\pi(0)} \vDash \gamma(\phi, \psi)$, and so by Lemma 3.6 since $x, b_{\pi(0)} \in \widetilde{\sigma}$ we have $\mathcal{X}, x \vDash \gamma(\phi, \psi)$.

Conversely, assume that $\mathcal{X}, x \vDash \gamma(\phi, \psi)$, which by Lemma 3.11 amounts to the existence of a PL-path $\pi : [0, 1] \overset{PL}{\to} P$ such that $\pi(0) = x$, $\pi(1) \in \llbracket\psi\rrbracket^{\mathcal{X}}$ and $\pi((0, 1)) \in \llbracket\phi\rrbracket^{\mathcal{X}}$. π is a simplicial path by Lemma 3.13, so there exist points $s_0 = 0 < s_1 < \cdots < s_l = 1$ such that each $\pi((s_i, s_{i+1}))$ is fully contained in a single cell. In particular, we can find cells $\widetilde{\tau}_1, \ldots, \widetilde{\tau}_l, \widetilde{\tau}_0', \ldots, \widetilde{\tau}_l'$ such that $\pi((s_{j-1}, s_j)) \subseteq \widetilde{\tau}_j$ for $j = 1, \ldots, l$, and such that $\pi(s_j) \in \widetilde{\tau}_j'$ for $j = 0, \ldots, l$. Notice that $\widetilde{\tau}_0' = \widetilde{\sigma}$, $\widetilde{\tau}_l' \subseteq \llbracket\psi\rrbracket^{\mathcal{X}}$ (since $\pi(1) \in \widetilde{\tau}_l'$) and $\widetilde{\tau}_i, \widetilde{\tau}_j' \subseteq \llbracket\phi\rrbracket^{\mathcal{X}}$ for $i = 1, \ldots, l$ and $j = 1, \ldots, l-1$ (since they contain points in $\llbracket\phi\rrbracket^{\mathcal{X}}$).

Observe that $\pi(s_j) \in \widetilde{\tau}_j' \cap \mathcal{C}(\widetilde{\tau}_{j+1})$ for $j = 0, \ldots, l-1$, and so by Lemma A.3 we have $\widetilde{\tau}_j' \overset{\sim}{\preceq} \widetilde{\tau}_{j+1}$. With a similar argument, we also have $\widetilde{\tau}_j' \overset{\sim}{\preceq} \widetilde{\tau}_j$ for $j = 1, \ldots, l$. Rewriting the previous conditions we have $\widetilde{\tau}_0' \overset{\sim}{\preceq} \widetilde{\tau}_1 \overset{\sim}{\succeq} \widetilde{\tau}_1' \overset{\sim}{\preceq} \widetilde{\tau}_2 \overset{\sim}{\succeq} \ldots \overset{\sim}{\succeq} \widetilde{\tau}_l'$, which by definition means that the sequence $\langle \widetilde{\tau}_0', \widetilde{\tau}_1, \widetilde{\tau}_1', \widetilde{\tau}_2, \widetilde{\tau}_2', \ldots, \widetilde{\tau}_l, \widetilde{\tau}_l' \rangle$ is a $\pm$-path of $\mathcal{M}(\mathcal{X})$. By previous considerations together with the inductive hypothesis applied to ϕ and ψ, we also have that $\widetilde{\tau}_0' = \widetilde{\sigma}$, that $\widetilde{\tau}_l' \in \llbracket\psi\rrbracket^{\mathcal{M}}$ and that $\widetilde{\tau}_i, \widetilde{\tau}_j' \in \llbracket\phi\rrbracket^{\mathcal{M}}$ for $i = 1, \ldots, l$ and $j = 1, \ldots, l-1$. Thus we have $\mathcal{M}(\mathcal{X}), \widetilde{\sigma} \vDash \gamma(\phi, \psi)$, as desired. $\square$

Theorem 6.3. *Logical equivalence is a simplicial bisimulation.*

Proof. In the proof, for S a set of logically equivalent points, we call "characteristic formula" of S the characteristic formula of the equivalence class that includes S; similarly, we also speak of the "characteristic formula" of a point x.

Consider two points x and y, with $x \equiv y$. Let us look at the conditions of Definition 6.2. First observe that, since ϕ can be an atomic proposition symbol, Condition 1 holds. We only prove Condition 2, as the proof of Condition 3 follows the same pattern.

Equivalently, we shall prove, by induction on k, the following statement: for each $k \geq 1$, for each pair x, y with $x \equiv y$, for each path π_x with $\pi_x(0) = x$, points $s_0, \ldots, s_k$, and cells $\widetilde{\sigma}_1, \ldots, \widetilde{\sigma}_k$ making π_x a simplicial path according to Definition 3.12, there is a simplicial path π_y with $\pi_y(0) = y$ and $\pi_x \widehat{\equiv} \pi_y$.

To ease readability, below, given the data above, we let $\phi_1, \ldots, \phi_k$ be the characteristic formulas of the sets $\pi_x((s_0, s_1)), \ldots, \pi_x((s_{k-1}, s_k))$ respectively. Moreover, let $\phi_0', \ldots, \phi_k'$ be the characteristic formulas of the points $\pi_x(s_0), \ldots, \pi_x(s_k)$ respectively.

Next, the proof proceeds by induction on k.

For $k = 1$, observe that, by Definition 3.3, we have $\mathcal{X}, x \vDash \gamma(\phi_1, \phi_1')$. By $x \equiv y$, we have $\mathcal{X}, y \vDash \gamma(\phi_1, \phi_1')$. By Lemma 3.11, there is a *piecewise linear* path π_y with $\pi_y(0) = y$, $\pi_y((0, 1)) \subseteq [\![\phi_1]\!]^{\mathcal{X}}$, and $\pi_y(1) \in [\![\phi_1']\!]^{\mathcal{X}}$. By Lemma 3.13, π_y is simplicial. Note that, since all the ϕ_i and ϕ_i' are characteristic formulas of equivalence classes, all the points in $\pi_x((0, 1)) \cup \pi_y((0, 1))$ are logically equivalent, and $\pi_x(1) \equiv \pi_y(1)$. Therefore, $\pi_x \hat{\triangleq} \pi_y$.

For $k > 1$, consider the sub-paths $\pi_x^a = \pi_x^{[0, s_1]}$ and $\pi_x^b = \pi_x^{[s_1, 1]}$. By Lemma A.4, both sub-paths are simplicial. By the previous case, there is a simplicial path π_y^a with $\pi_x^a \hat{\triangleq} \pi_y^a$. In particular, we have $\pi_x^a(1) \equiv \pi_y^a(1)$. Noting that $\pi_x^a(1) = \pi_x(s_1) = \pi_x^b(0)$, we now apply the inductive hypothesis to the points $\pi_x^b(0)$, $\pi_y^a(1)$ and the simplicial path π_x^b, obtaining the simplicial path π_y^b with $\pi_y^b(0) = \pi_y^a(1)$ and $\pi_x^b \hat{\triangleq} \pi_y^b$. Let $\pi_y = \pi_y^a; \pi_y^b$. By Lemma A.4, π_y is simplicial, and we have $\pi_x = \pi_x^a; \pi_x^b \hat{\triangleq} \pi_y^a; \pi_y^b = \pi_y$, proving the thesis. $\qquad\square$

Theorem 6.4. *Each simplicial bisimulation is included in logical equivalence.*

Proof. Given a simplicial bisimulation $\sim$, we need to show that for every formula ϕ and all points x and y, if $x \sim y$ then $\mathcal{X}, x \models \phi \iff \mathcal{X}, y \models \phi$. We use induction on the structure of ϕ. The cases for atomic propositions and Boolean operations are trivial, thus omitted. The case for the $\square$ operator is also omitted, as $\square$ is derived from γ by Theorem 3.8.

Suppose $x \sim y$ and $\mathcal{X}, x \vDash \phi = \gamma(\phi_1, \phi_2)$. By Lemma 3.11, there is a piecewise linear path π_x with $\pi_x(0) = x$, $\pi_x((0, 1)) \subseteq [\![\phi_1]\!]^{\mathcal{X}}$, and $\pi_x(1) \in [\![\phi_2]\!]^{\mathcal{X}}$. By Lemma 3.13, π_x is simplicial. By Condition 2 of Definition 6.2, and the fact that $x \sim y$, there is a simplicial path π_y with $\pi_y(0) = y$, such that $\pi_x \hat{\sim} \pi_y$. For each $r \in (0, 1)$, we have $\pi_x(r) \sim \pi_y(r)$ and $\mathcal{X}, \pi_x(r) \models \phi_1$. By induction hypothesis, $\mathcal{X}, \pi_y(r) \vDash \phi_1$, therefore $\pi_y((0, 1)) \subseteq [\![\phi_1]\!]^{\mathcal{X}}$. We also have $\mathcal{X}, \pi_x(1) \vDash \phi_2$ and $\pi_x(1) \sim \pi_y(1)$. Thus, by induction hypothesis, $\pi_y(1) \in [\![\phi_2]\!]^{\mathcal{X}}$. Summing up, we have $\mathcal{X}, y \vDash \gamma(\phi_1, \phi_2)$, concluding one direction of our proof. Next, we should prove that conversely, if $\mathcal{X}, y \vDash \gamma(\phi_1, \phi_2)$ then $\mathcal{X}, x \vDash \gamma(\phi_1, \phi_2)$. The argument is similar to the other case, using Condition 3 of Definition 6.2 instead of Condition 2. $\qquad\square$