Spatio-Temporal Predicates

Martin Erwig and Markus Schneider

Abstract—This paper investigates temporal changes of topological relationships and thereby integrates two important research areas: First, two-dimensional topological relationships that have been investigated quite intensively and, second, the change of spatial information over time. We investigate spatio-temporal predicates, which describe developments of well-known spatial topological relationships. A framework is developed in which spatio-temporal predicates can be obtained by temporal aggregation of elementary spatial predicates and sequential composition. We compare our framework with two other possible approaches: one is based on the observation that spatio-temporal objects correspond to three-dimensional spatial objects for which existing topological predicates can be exploited. The other approach is to consider possible transitions between spatial configurations. These considerations help to identify a canonical set of spatio-temporal predicates.

Index Terms—Time in geographic information, spatio-temporal data types, representation of spatio-temporal objects, changes of spatial predicates, developments of spatial objects.

1 Introduction

The affinity of spatial and temporal phenomena has been recognized for a long time in literature. Both phenomena deal with "spaces" or "dimensions" of some kind and are thus closely related. The (regained) awareness of their deep relationships has led, both in spatial and in temporal data modeling, to an increased interest in integrating both directions into a common research branch called *spatio-temporal data modeling* and in constructing *spatio-temporal data bases*. Their underlying basic entities are called *spatio-temporal objects* and are ubiquitous in everyday life. Consider the flight of an airplane, the migration of whales, the raging of a storm, or the spreading of a fire region. Characteristic features of all these objects are that they are *spatial entities changing over time* and that these changes are *continuous*. Changes refer to the motion, shrinking, growing, shape transformation, splitting, merging, disappearing, or reappearing of spatio-temporal objects. In particular, the capability of incorporating *continuous* change of spatial objects over time belongs to the most challenging requirements of spatio-temporal data modeling. In the meantime, some spatio-temporal data models (which will be described later) have already been proposed.

Temporal changes of spatial objects induce modifications of their mutual topological relationships over time. For example, at one time, two spatio-temporal objects might be disjoint whereas, some time later, they might intersect. These modifications usually proceed continuously over time but can, of course, also happen in discrete steps. Currently, a still open issue, which will be the main subject of this paper, relates to the nature and formal definition of these *spatio-temporal relationships* which will be described by so-called *spatio-temporal predicates*.

- *M. Erwig is with the Department of Computer Science, Oregon State University, Corvallis, OR 97331. E-mail: erwig@cs.orst.edu.*
- *M. Schneider is with Praktische Informatik IV, Fern Universität Hagen, 58084 Hagen, Germany. E-mail: markus.schneider@fernuni-hagen.de.*

Manuscript received 23 Dec. 1999; revised 2 Feb. 2001; accepted 5 Feb. 2001. For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 111124.

A brief motivating example shall illustrate our modeling approach for these predicates. Consider a database containing information about the flights of airplanes and about weather conditions. The query whether an airplane *crossed* a certain storm has a spatio-temporal nature and means to check the validity of different spatial predicates during a series of events and periods in a given temporal order. This means we have to examine whether there has been a constellation when the plane and the storm were disjoint for a while, when they met at some time, when the plane was inside the storm for a while, when the plane again reached the border of the storm at some time, and, finally, when the plane and the storm were disjoint again. We can observe that, during certain time periods, the topological relationships between both objects are constant and that, at certain points in time, these relationships change. Consequently, we obtain an alternating sequence of time intervals and time points at which the topological relationships between both objects are constant. Roughly speaking, a spatio-temporal relationship is a sequence of (well-known) spatial relationships that hold over time intervals or at time points; we will call it a *development*.

The paper also addresses two other related issues. The first one refers to the existence of a canonical collection of spatio-temporal predicates from which more complex ones can be constructed. The second issue is based on the observation that spatio-temporal objects can be regarded as *three-dimensional geometric objects* (for example, the temporal evolution of a region can be considered as a 3D volume). We are interested in comparing the expressiveness of topological relationships for 3D objects with developments for spatio-temporal objects.

In Section 2, we mention our own previous work, discuss related work, and identify further objectives of this paper. In a more formal way, Section 3 explains the foundations and assumptions for a design of spatio-temporal objects and predicates. In Section 4, the nature of spatio-temporal predicates is explained and formally defined. Temporal quantifiers enable a definition of basic spatio-temporal predicates. We also show how more

complex spatio-temporal predicates can be composed, for example, as sequences of already existing predicates. Indeed, we present a rather complete set of combinators and prove many of their properties. Section 5 compares topological predicates on spatio-temporal objects with topological predicates on 3D objects. Section 6 attempts to identify a canonical collection of spatio-temporal predicates. Finally, Section 7 concludes the paper.

2 Related Work

In this section, we will briefly review our own and related work, as far as it is relevant for this paper. Our design of spatio-temporal predicates requires knowledge from several fields. First, a treatment of these predicates necessitates an understanding of the underlying spatio-temporal objects. In Section 2.1, we will therefore deal with spatio-temporal data models introduced so far. There we will also sketch our own approach. Second, for modeling spatio-temporal predicates, we can learn much from spatial data modeling. Section 2.2 will mention well-known work on topological predicates for spatial objects and outline how it can be exploited for our purposes. In a certain sense, we start from this spatial or topological domain and appropriately add the temporal aspect to model spatio-temporal predicates. Third, we investigate in Section 2.3 to which extent logic-based approaches have been applied to model spatio-temporal predicates; in particular, we discuss why we have deliberately chosen an alternative approach.

2.1 Spatio-Temporal Objects

So far, only a few data models for spatio-temporal data have been proposed. One approach has been to appropriately extend a spatial data model by temporal concepts. In [41], spatio-temporal objects are defined as so-called spatio-bitemporal complexes. Their spatial features are described by simplicial complexes, and their temporal features are given by bitemporal elements attached to all components of simplicial complexes. In [39], in a discrete snapshot model, a spatio-temporal object o is given as a time-evolving spatial object, that is, its evolution is represented by a set of triples (o_id, s_i, t_i), where o_id is the object identifier of o and where s_i is the location of o at instant t_i. Another approach has been to appropriately extend a temporal data model based on attribute time-stamps by spatial concepts. In contrast to tuple time-stamped models, attribute time-stamped models (described in [8], [9], [24], [34], for example) aim at gathering information about an object in one tuple and allow complex attribute values. These complex values incorporate the temporal dimension and are frequently modeled as functions from time into a value domain, a view which is quite similar to ours. From this perspective, attribute time-stamped models have the potential to fit very well with our purposes. Examples of temporal models integrating spatial aspects include variants of Gadia's temporal model [24], which are described in [6], [5]. A third approach has been to use linear constraints for modeling spatio-temporal data [27]. This model allows us to efficiently represent and manipulate infinite point sets in arbitrary dimension. But, in [27], time and geometry are treated as different and independent categories of data. Unfortunately, this impedes the modeling of continuous evolutions which is in principle possible in that model if we consider time as an additional geometric dimension. In summary, the main drawback of all approaches mentioned so far is that, ultimately, they are currently incapable of modeling *continuous* changes of spatial objects over time.

Our approach to dealing with spatio-temporal data supports a more integrated view of space and time and incorporates the treatment of continuous spatial changes. It will be the basis of this paper and introduces the concept of *spatio-temporal data types* [15], [16]. These data types are designed as abstract data types whose values can be integrated as complex entities into databases [36], [35] and whose definition and integration into databases is independent of a particular DBMS data model.

The definition of a temporal object [22] in general is motivated by the observation that anything that changes over time can be expressed as a function over time. A temporal version of an object of type α is then given by a function from *time* to α. Spatio-temporal objects are regarded as special instances of temporal objects where α is a spatial data type like *point* or *region*. A point (representing an airplane, for example) that changes its location in the Euclidean plane over time is called a *moving point*. Similarly, a temporally changing region (representing a fire area, for example) is a region that can move and/or grow/shrink. We call such an object an *evolving region*. Spatio-temporal objects can even disappear or reappear and, for an evolving region, its components can either split or merge. A more detailed and more formal overview of spatio-temporal data types will be given in Section 3.3.

A straightforward and very instructive view of spatio-temporal objects is to visualize their temporal evolution as purely geometric, *three-dimensional* objects, that is, the time axis is regarded as an additional third geometric dimension. An evolving region is then represented as a *volume* in 3D space, and a moving point is then visualized as a *3D curve* such that any intersection parallel to the *xy*-plane yields a spatial object, that is, a region or a point. We will use this view for a comparison of topological predicates on spatio-temporal objects with those on 3D objects (see Sections 2.2, 3.2, and 5).

Similar to our approach, in [42], [43], so-called *behavioral time sequences* have been introduced which are based on the work in [34]. Each element of such a sequence contains a geometric value, a date, and a *behavioral function*; the latter describes the evolution between two consecutive elements of the sequence. Whereas this approach mainly focuses on representational issues and advocates the *three-dimensional object view* of spatio-temporal objects, we are particularly interested in an algebraic model of general spatio-temporal data types, including a comprehensive collection of spatio-temporal operations [16]. Nevertheless, behavioral time sequences could be used as one possible representation for our temporal objects.

Galton [25], [26] stresses continuity as an important feature of movement. His goal is to devise a framework for formalizing common-sense knowledge of the world, and he particularly focuses on how continuity fits into a qualitative

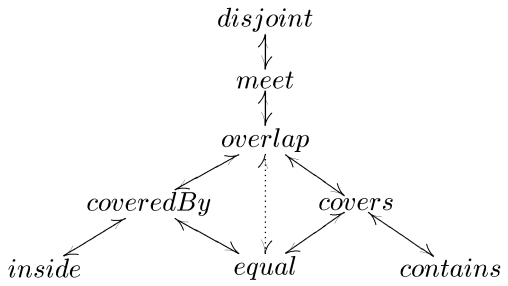

Fig. 1. Conceptual neighborhood graph for the region/region case.

setting. Based on different separation measures for regions, he identifies different kinds of continuity.

In this paper, our discussion of spatio-temporal objects focuses on the temporal evolution of single, self-contained spatial objects. One could also take into account collections of temporally evolving spatial objects possibly satisfying some constraints over time. For example, an approach to modeling *spatio-temporal partitions* for describing the *temporal evolution of maps* has already been given in [19].

2.2 Spatio-Temporal Predicates

In the past, topological relationships like *meet*, *overlap*, or *inside* between point, line, and region features in the (especially two-dimensional) Euclidean space have been intensively studied in the literature. Using point sets and point set topology as a formal framework, the 9-intersection model [13], [7] provides a canonical collection of topological predicates for each combination of spatial types. Based on the nine topologically invariant intersections of boundaries, interiors, and exteriors of the two participating objects, these predicates are mutually exclusive and cover each possible topological situation between two objects.

We exploit this model in two ways. First, we regard spatio-temporal objects as 3D objects and compare the expressive power of the topological predicates (Egenhofer's 9-intersection model) on 3D objects with the spatio-temporal predicates on spatio-temporal objects (Section 5). Second, we are interested in designing a canonical collection of spatio-temporal predicates (Section 6). A first hint to a possible design is presented in the work in [12] which considers *possible topological transitions* (that is, changes) between topological relationships that are represented in a so-called "conceptual-neighborhood-graph" or "closest-topological-relationship-graph" (Fig. 1). For example, if two regions are disjoint and later they overlap, then there must have been a topological situation where both regions have met each other. Although one can have time in mind, a temporal component is missing in the graph; the graph is purely spatial.

The concepts for spatio-temporal predicates and developments presented in the following sections have already been used in two different but related applications. The first application described in [18] relates to the identification of an important new class of *spatio-temporal queries* which is concerned with developments of spatial objects over time. This means queries ask especially for *changes in spatial relationships over time*. A macromechanism is provided which allows the user to build more and more complex spatio-temporal predicates starting with a small set of elementary ones. We have demonstrated how these concepts can be realized in a relational model and how SQL can be appropriately extended to a spatio-temporal query language called *STQL* to enable the querying of developments.

The second application described in [20], [21] deals with the user's problem of comfortably specifying temporally changing topological situations and predicates. For this purpose, we have proposed a *visual language* which extends existing concepts for visual spatial query languages that are only capable of querying *static* topological situations. The paper motivates the language design and describes the translation process into our formal model of spatio-temporal predicates. The visual notation can be used directly as a visual query interface to spatio-temporal databases, or it can provide predicate specifications that can be integrated into textual query languages leading to heterogeneous languages [17].

2.3 Logic-Based Approaches

An obvious approach for dealing with temporal predicates is to employ concepts of temporal logic. We will first mention some related work and afterward explain why we will take a partly different approach in this paper.

Based on his work in [1] where he has identified 13 possible relationships (like *before*, *equal*, *meets*, *overlaps*, *during*) between intervals, Allen in [2] defines a predicate $Holds(p, i)$ which asserts that a property p is true during a time interval i. This predicate implies that p holds at every subinterval of i as well. Logical operators like Boolean connectives and quantifiers allow us to combine several *Holds* predicates. What we call development is here called *event*. A predicate $Occurs(e, i)$ is introduced which takes an event e and a time interval i and only yields true if the event applies to the entire interval i, but not to any proper subinterval of i. In both papers, Allen's temporal logic is solely based on time intervals and not on time points.

Galton [25] has extended Allen's approach to the treatment of temporally changing topological relationships. Topological relationships are based on the RCC model [10], [31] which comes to similar results as Egenhofer's 9-intersection model. In contrast to Allen, Galton also takes time points into account, as we do. In specifying changes of spatial situations, he uses the notion of a *fluent* or *state* which corresponds to what we call a temporal function. A fluent can have different values at different times. Besides the $Holds(S, i)$ predicate which indicates that fluent S holds throughout the interval i, Galton also introduces a predicate $Holds\text{-}at(S, t)$, which indicates that S holds at instant t. If S denotes a spatial predicate, this predicate can be valid for i or at t, respectively. Since movement is not a fluent but an event, the predicates $Occurs(e, i)$ and $Occurs\text{-}at(e, t)$ are introduced; they indicate that an event of type e occurs over an interval i or at an instant t, respectively. For example, Galton formulates the query whether a region a enters a region r as follows (topological predicates are given in our notation, $Beg(i)$ and $End(i)$ denote the begin and end point of the open interval i):

$$Occurs(enter(a,r),i) := Holds\text{-}at(meet(a,r),Beg(i)) \wedge Holds(overlap(a,r),i) \wedge Holds\text{-}at(coveredBy(a,r),End(i)).$$

Whereas Galton takes into account time points, he does not consider geometric points. For two reasons, we have included time points in our model, too. First, we achieve symmetry to the existence of points in the spatial domain. Second and more importantly, we are enabled to project spatio-temporal values by function application at time points to spatial values. This feature is absolutely essential and much needed in spatio-temporal query languages.

The concept for spatio-temporal predicates presented in this paper is a hybrid approach taking into account elements from temporal logic (similar to the work in [3]) and elements from point set theory and point set topology. The main reason for not taking a purely logic approach is the intended integration into spatio-temporal databases and query languages. These require concrete representations for spatio-temporal objects and, besides predicates, the possibility to construct new objects from existing objects through spatio-temporal operations. Therefore, efficiency, especially for the evaluation of spatio-temporal queries, is indispensable. The evaluation process of queries can be based on 3D methods from Computational Geometry. Logical approaches cannot give efficient support for these requirements since they do not have concrete concepts and representations of objects; these are inherent features of the logical model. For all these reasons, we propose a concrete model and pursue a systematic study of spatio-temporal predicates which has not been a subject of the logical approaches mentioned above.

3 Foundations of Spatio-Temporal Data Types

In this section, we briefly recapitulate three fundamental concepts that are essential for an understanding of our approach to spatio-temporal predicates. Section 3.1 discusses basic and *spatial data types*. Section 3.2 recalls *topological predicates* on spatial objects, and Section 3.3 sketches the concept of *spatio-temporal data types*, which is our approach to modeling the continuous temporal evolution of spatial objects.

3.1 Basic and Spatial Data Types

We assume a standard interpretation of basic data types. However, we require that each domain is extended by a special value, $\perp$, denoting the "undefined" value. This extension will be later needed when so-called "lifted" spatial predicates are explained (see Section 3.3). Consequently, the Boolean type is defined as $bool = \mathbb{B} \cup \{\perp\}$, where $\mathbb{B} = \{T, F\}$. The other basic types for reals, integers, and strings are defined and extended analogously but not needed in this paper.

Spatial data types like *point*, *line*, and *region* have turned out to be a fundamental abstraction for modeling the two-dimensional structure of geometric entities, their properties, relationships, and operations (see, for example, [28], [30], [32], [33], [37]). Points are elements of the Euclidean plane. Lines are two-dimensional curves. Regions describe point sets with a two-dimensional extent and are bounded by lines which in this context are called *boundaries*.

These informal explanations indicate that our formal definition of spatial data types is based on point set theory in contrast to "pointless" theories (in the sense of not being based on point sets) like logic-based approaches. The reasons for this design decision have especially been the integration of spatial data types into databases, which necessitates concrete representations, and efficiency required for storing, retrieving, manipulating, and querying spatial objects.

In our context, type definitions also include the $\perp$ element. The definition of a type for points is simple: $point = \mathbb{R}^2 \cup \{\perp\}$. For regions, unfortunately, the use of pure point set theory causes problems. If regions are simply modeled as arbitrary point sets, they can suffer from undesired geometric anomalies. These degeneracies relate to isolated or dangling line and point features, as well as missing lines and points in the form of cuts and punctures. They all result either from erroneous object definitions or from operations on objects. But, the expectation in spatial applications is usually that spatial objects are in a certain sense well-defined and *regular*.

In order to be able to formally express this requirement and to avoid the anomalies just described, the formalization of a data type for regions is based on *point set topology* [23] and on a process called *regularization* [40]. Point set topology also rests on the point set paradigm but additionally distinguishes different parts of an (arbitrary) point set. Given such a point set, say A, these parts identify its *boundary* ∂A, its *interior* A°, and its *exterior* A^-, which are pairwise disjoint. The union of A° and ∂A corresponds to the *closure* $\overline{A}$ of A. The effect of applying the *interior* operation to a point set is to eliminate dangling points, dangling lines, and boundary parts. The effect of the *closure* operation is to eliminate cuts and punctures by appropriately supplementing points as well as adding the boundary. The concept of regularity defines a point set A as *regular closed* if $A = \overline{A^\circ}$, that is, the composition of the *interior* and the *closure* operation is applied to A and yields A. Consequently, it makes sense to define a *regularization function reg* which associates a set A with its corresponding regular closed set as $reg(A) := \overline{A^\circ}$.

We can now give a proper definition of a type for regions:

$$region = \{R \subseteq \mathbb{R}^2 |\ R \text{ is regular closed}\} \cup \{\perp\}.$$

A definition of a type for lines is omitted here since, in this paper, we will not consider their temporal versions.

The concept of regularity is independent of the dimension of the spatial objects. Hence, we can employ it, in particular, for spatial objects in the three-dimensional space. This is needed in Section 5.

3.2 Topological Predicates

Topological predicates between spatial objects in the two-dimensional space belong to the most investigated topics of spatial data modeling and reasoning. As mentioned in Section 2.2, we employ the 9-intersection model from which

TABLE 1
The Eight Topological Predicates for Two Regions/Volumes

$$\begin{pmatrix} F & F & T \\ F & F & T \\ T & T & T \end{pmatrix} \quad \begin{pmatrix} T & F & T \\ F & F & T \\ T & T & T \end{pmatrix} \quad \begin{pmatrix} T & T & T \\ T & T & T \\ T & T & T \end{pmatrix} \quad \begin{pmatrix} T & F & F \\ F & T & F \\ F & F & T \end{pmatrix}$$

disjoint *meet* *overlap* *equal*

$$\begin{pmatrix} F & T & F \\ F & T & F \\ T & T & T \end{pmatrix} \quad \begin{pmatrix} F & F & T \\ T & T & T \\ F & F & T \end{pmatrix} \quad \begin{pmatrix} T & F & T \\ T & T & T \\ F & F & T \end{pmatrix} \quad \begin{pmatrix} T & T & F \\ F & T & F \\ T & T & T \end{pmatrix}$$

inside *contains* *covers* *coveredBy*

a canonical collection of topological relationships can be derived for each combination of spatial types. The model is based on the nine possible intersections of boundary, interior, and exterior of a spatial object with the corresponding parts of another object. Each intersection is tested with regard to the topologically invariant criteria of emptiness and nonemptiness. This can be expressed for two spatial objects A and B by evaluating the following matrix:

$$\begin{pmatrix} \partial A \cap \partial B \neq \emptyset & \partial A \cap \partial B^\circ \neq \emptyset & \partial A \cap \partial B^- \neq \emptyset \\ A^\circ \cap \partial B \neq \emptyset & A^\circ \cap B^\circ \neq \emptyset & A^\circ \cap B^- \neq \emptyset \\ A^- \cap \partial B \neq \emptyset & A^- \cap B^\circ \neq \emptyset & A^- \cap B^- \neq \emptyset \end{pmatrix}.$$

For this matrix, $2^9 = 512$ different configurations are possible from which only a certain subset makes sense, depending on the combination of spatial objects just considered. For two regions, eight meaningful configurations [13] have been identified which lead to the eight predicates called *equal*, *disjoint*, *coveredBy*, *covers*, *overlap*, *meet*, *inside*, and *contains*. For a line and a region, 19 topological relationships [14] can be distinguished. For two lines, 33 relationships [11] can be found. For a point and a region, we obtain the three predicates *disjoint*, *meet*, and *inside*. For two points, we get the two predicates *disjoint* and *meet* (which corresponds to equality). For each group, this means that each of its predicates is associated with a unique intersection matrix (see Table 1 and references mentioned above) so that all predicates are mutually exclusive and complete with regard to the topologically invariant criteria of emptiness and nonemptiness.

All these topological predicates are restricted in the sense that their argument objects are not allowed to be arbitrary but must, in particular, satisfy two constraints. First, each spatial object may only consist of exactly one component, that is, each object is connected. Second, regions are regular closed sets of $\mathbb{R}^2$ (see Section 3.1) and are not permitted to contain holes. Since the topological predicates are also applicable to higher dimensions and especially to the third dimension, the restrictions on the objects in the two-dimensional space also have consequences for the corresponding spatial objects in the three-dimensional space. Single two-dimensional points then correspond to single three-dimensional lines or curves, and single two-dimensional regions without holes correspond to single three-dimensional volumes without holes (see Section 5).

Possible *topological transitions* (that is, changes) between topological relationships can be represented in a so-called *conceptual neighborhood graph* or *closest-topological-relationship-graph* [12]. Topological changes occur if the topological relationship between two spatial objects is affected by reshaping or moving one or both objects. Each set of topological relationships concerning the point/point, point/region, line/line, line/region, and region/region cases can be organized in such a graph so that similar relationships are close to each other. In such a graph, the relationships are the vertices and the conceptual neighborhood is given by edges between vertices. In the case of region/region relationships, the conceptual neighborhoods are identified by the so-called *topological distance* [12] which, for the 9-intersection matrices of any two distinct relationships, is calculated as the number of different T and F entries. Pairs of relationships with the least nonzero number of differences are considered to be conceptual neighbors. Fig. 1 shows the conceptual neighborhood graph for two regions. An essential feature of this graph is that each of its paths only incorporates a sequence of snapshots of changing topological relationships. Starting from a vertex v, all incident edges lead to the *next possible* topological relationships and, thus, indicate immediately possible qualitative transitions. All other relationships that are not connected to v cannot be directly reached from v. At this point, it should be mentioned that one special case is ignored by the graph due to its construction procedure. In the case that both regions have the same shape and size, a direct transition from *equal* to *overlap*, and vice versa, is possible. Thus, from this perspective, an edge between the corresponding nodes in the graph could be added.

Time only implicitly plays a role here and is not necessarily important, although helpful, for an understanding of the graph. This is, of course, not astonishing since this graph only deals with spatial and not with temporal behavior.

Due to the larger numbers of topological predicates for the line/region and the line/line cases, the corresponding conceptual neighborhood graphs are more complex and not presented here. For these two cases, the reader is referred to [11], [14]. For the point/point case and the point/region case, the graphs are obvious.

3.3 Spatio-Temporal Data Types

In this section, we outline the data model underlying spatio-temporal predicates. This first requires a suitable concept of time. For compatibility with smoothly changing spatio-temporal objects, we choose a continuous model of time, that is, $time = \mathbb{R}$. In terms of temporal databases, this is a simple but, for our purposes, sufficient concept of valid time. Next, to model values of an atomic, nontemporal type α that change over time, we introduce the notion of *temporal function*, which is an element of type

$$\tau(\alpha) = time \to \alpha.$$

Type $\tau(\alpha)$ denotes all total functions from time to α. We say $\omega : \tau(\alpha)$ is defined at time t if $\omega(t) \neq \perp$, and we define the *domain* of ω as $dom(\omega) = \{t \mid \omega(t) \neq \perp\}$. Note that $\tau(\alpha)$ itself does not include $\perp$; thus, a completely undefined temporal object is represented by the function that maps each time to $\perp$. The introduction of the special symbol $\perp$ allows us to comfortably model situations where temporal objects disappear or reappear.

A temporal function $\omega : \tau(\alpha)$ has to satisfy some constraints. First, we require that it is *upper semicontinuous*. Consider the temporal development of Germany. For over 40 years, its area was constant, then changed at some time t (reunification of West and East Germany) and has since then been constant. What was Germany's area at time t? Obviously, we cannot come to an objective decision, but only know that not both the old and the new area can simultaneously belong to time t. We have to decide arbitrarily and to assign one of the two areas to it. This especially maintains the functional character of our temporal object view. We have chosen to ascribe the temporally later value to such a *temporal change point*. For our example, this implies that Germany was reunified and did not remain separated at time t. This means that time intervals describing the lifespan of a temporal object have to be right half-open. Upper semicontinuity is the mathematical concept for defining this constraint. Intuitively, it means that ω has to be continuous from the upper side. If ω is continuous, then it is also upper semicontinuous. For the case that ω is a spatio-temporal object, we have given formal definitions of this concept which are based on "difference measures" for spatial objects [19], [22].

Second, as a direct conclusion, the time of ω's evolution after a temporal change point must be an interval up to the next temporal change point and is not allowed to be a single time point. That is, there are no "thin, isolated temporal values" at temporal change points. From an application point of view, this has to be regarded as an anomaly. Consider again the result of the reunification of West and East Germany, which, after that event until now, has lasted for a time period and not only for a time point. Third, the number of continuity gaps, which relate to periods where ω is undefined, has to be finite in order to be able to represent (spatio-)temporal objects. Further explanations to restrictions of type τ that ensure representation and computability are discussed in [22].

We have employed temporal functions as the basis of an algebraic model for *spatio-temporal data types* where α is assigned one of the spatial data types *point*, *line*, or *region*. The results are *moving points* as values of type τ(*point*), *evolving lines* as values of type τ(*line*), and *evolving regions* as values of type τ(*region*). A moving point is a point that changes its location over time. An evolving line is a line that can move and/or grow/shrink and/or change its linear structure. Similarly, an evolving region is a region that can move and/or grow/shrink and/or change its areal extent. Due to space constraints and to keep the model from becoming overly complex, in this paper, we exclusively focus on moving points and evolving regions; similar considerations could, of course, also be done for evolving lines.

In addition, we also have changing numbers or Booleans, which are essential when defining operations on temporal objects. For instance, we could be interested in computing the (time-dependent) distance of an airplane and a storm. This could be achieved by an operation:

$$Distance : \tau(point) \times \tau(region) \to \tau(real).$$

This example demonstrates the important principle of *temporal lifting* [29] in avoiding an inflation of operation definitions: We can, in principle, take almost any flat (that is, nontemporal) operation and "lift" it so that it works on temporal objects returning also a temporal object as a result. More precisely, for each (flat) function $f : \alpha_1 \times \ldots \times \alpha_n \to \beta$, its corresponding lifted version is defined by:

$$\uparrow f : \tau(\alpha_1) \times \ldots \times \tau(\alpha_n) \to \tau(\beta)$$

with

$$\uparrow f(S_1, \ldots, S_n) := \{(t, f(S_1(t), \ldots, S_n(t))) \mid t \in time\}.$$

Thus, we can derive temporal operations rather automatically. For example, we have $Distance = \uparrow distance$, which is the lifted version of the operation

$$distance : point \times region \to real.$$

In order to make notations more discernible, we use the following conventions: We denote nontemporal, "flat" entities and operations by lowercase letters, whereas temporal objects start with capital letters. Hence, we shall denote, for example, a point by p and a region by r. In contrast, a moving point and an evolving region are denoted by P and R, respectively. The same applies to functions and types: The spatial operation *distance* takes (for example) objects of type *region* and *point* and computes a number of type *real*, whereas its lifted version $Distance = \uparrow distance$ maps elements of type $Region = \tau(region)$ and $Point = \tau(point)$ (that is, evolving regions and moving points) to $Real = \tau(real)$ (temporal reals). In contrast to these naming conventions, predicate names starting with a capital letter are used to denote spatio-temporal predicates (which map spatio-temporal objects to $bool \setminus \{\perp\}$, that is, $\mathbb{B}$).

For more operations on spatio-temporal objects and, in particular, for many example queries, see [16].

4 Predicates on Spatio-Temporal Objects

In this section, we define spatio-temporal predicates and investigate many of their properties. After scrutinizing the nature of spatio-temporal predicates in Section 4.1, we consider different aggregation operators to combine Boolean values that change over time in Section 4.2. These are used in Section 4.3 to define a set of basic spatio-temporal predicates.

How these can be combined into more complex predicates is shown in Section 4.4, and a pleasing syntactical notation for predicate composition is defined in Section 4.5. More predicate combinators are then presented in Section 4.6, and possibilities for the definition of nonbinary predicates are explained in Section 4.7. A number of illustrating examples follows in Section 4.8 and, finally, Section 4.9 collects useful and interesting laws about spatio-temporal predicates.

4.1 What Are Spatio-Temporal Predicates?

We can easily apply the idea of temporal lifting to spatial predicates: Since a spatial predicate is a function from spatial objects to *bool*, a temporally lifted spatial predicate is a function from spatio-temporal objects to temporal Booleans (that is, *Bool*). The essence of predicates is, however, that they can be used to express facts which can be either true or false. Thus, we have to aggregate (for example, by temporal quantifiers) the values along the time axis into a single Boolean. Note that whereas a lifted predicate might well return $\perp$, this cannot happen for a spatio-temporal predicate; instead, undefined values are aggregated into Booleans. Accordingly, we can define:

Definition 1 (Spatio-Temporal Predicate). *A spatio-temporal predicate is a function of type* $\tau(\alpha) \times \tau(\beta) \to \mathbb{B}$ *for* $\alpha, \beta \in \{point, region\}$.

Let us consider some examples. The spatial predicate *inside* is defined, for example, for points and regions, that is,

$$inside : point \times region \to bool.$$

The lifted version of this predicate has the following type:

$$\uparrow inside : Point \times Region \to Bool$$

with the meaning that it yields true for each time point at which the moving point is inside the evolving region; it yields undefined whenever the point or the region is undefined, and it yields false in all other cases. Finally, we can define a spatio-temporal predicate *always-inside* that yields true if and only if $\uparrow inside$ yields true for all times. However, this definition is a bit problematic, since it yields false whenever the point, the region, or both are undefined. This is a very restrictive view, which also is not intuitive in certain cases (for example, when the moving point is inside the evolving region during its whole lifetime). In the next section, we therefore investigate the definition of temporal quantifiers in some detail.

The above definition captures only the case of binary spatio-temporal predicates. In fact, we are mainly interested in these, but we can easily extend the concept to predicates of arbitrary arity. For example, a unary spatio-temporal predicate is a function of type $\tau(\alpha) \to \mathbb{B}$ (for $\alpha, \beta \in \{point, region\}$). In Section 4.7, we will consider unary predicates in some detail.

4.2 Temporal Aggregation

A lifted spatial predicate yields a defined value only on the intersection of the domains of two spatio-temporal objects. This affects the definition of aggregate operators, such as "$\forall$" and "$\exists$": We have to decide how "strict" these quantifiers have to be, that is, do we map undefined values to true or to false and do we aggregate only over a restricted time interval (for example, the intersection of the two participating temporal objects' domains)?

We overload the notation of quantifiers from logic and denote universal aggregation by $\forall$ and existential aggregation by $\exists$. Both operators take a spatial predicate and deliver a spatio-temporal predicate, that is, they both have the following (higher order) type:

$$(\alpha \times \beta \to bool) \to (\tau(\alpha) \times \tau(\beta) \to \mathbb{B}).$$

The semantics of existential quantification presents no difficulties. Let p be a spatial predicate, and let S_1 and S_2 be two spatio-temporal objects. Then, $\exists p(S_1, S_2)$ is true if and only if p is true for the values of S_1 and S_2 at some time. (In the following, we always assume that t ranges over *time* unless stated otherwise.)

$$\exists p \ := \ \lambda(S_1, S_2).\exists t : p(S_1(t), S_2(t)).$$

A note on the lambda-notation: $\lambda(x_1, \ldots, x_n).e$ denotes a function that takes arguments $x_1, \ldots, x_n$ and returns a value determined by the expression e. When this function is applied to arguments $a_1, \ldots, a_n$, the body e is evaluated with (all free occurrences of) x_i substituted by a_i.

In contrast to $\exists$, we have several possibilities for defining the meaning of $\forall$ depending on the time interval over which the universal quantification is assumed to range. This means, if t_1 and t_2 are the respective domains of the two spatio-temporal objects S_1 and S_2, the result of $\forall p(S_1, S_2)$ depends, in general, on whether quantification ranges over

1. *time*,
2. $t_1 \cup t_2$,
3. t_1,
4. t_2, and
5. $t_1 \cap t_2$.

The first case is the most restrictive one: $\forall p$ can be true only if both objects are totally defined. The second option is also very restrictive since it actually demands both arguments to have the same domain. This could be intended, for example, if we ask whether two countries have always been neighbors. This means the *meet* predicate has to be true for all times any of the two countries exists, which can be true only if the two countries have the same lifetime. Cases 3 and 4 require $t_1 \subseteq t_2$ and $t_2 \subseteq t_1$, respectively. An example for Case 3 is: "Did a certain species (always) live in a particular climate region?" This requires the *inside* predicate to be true during the existence of the species: It does not matter when the lifetime of the climate region properly includes that of the species, but, if the species existed before or after the region, the spatio-temporal predicate would be expected to yield false. (Case 4 is just the dual case of 3.) The last case is least restrictive; it just demands the predicate to hold on the two object's common lifetime, in particular, if $t_1 \cap t_2 = \emptyset$, $\forall p$ yields trivially true. An example is: "Was a certain flight able to avoid a storm?" This is particularly true at those times when one or the other object does not exist.

The first case describes a generally unrealistic assumption, so we do not consider this alternative any further. As we have seen, all other cases do make sense, and we obtain definitions of four different quantifiers. Hence, Cases 2 to 4 are captured

by the quantifiers $\forall_\gamma$, where $\gamma \in \{\cup, \cap, \pi_1, \pi_2\}$ and where $\pi_i(x_1, \ldots, x_i, \ldots, x_n) = x_i$. These quantifiers are defined as follows:

$$\forall_\gamma p \ := \ \lambda(S_1, S_2).\forall t \in \gamma(dom(S_1), dom(S_2)) : p(S_1(t), S_2(t)).$$

Finally, we use the following abbreviating syntax for operators that create spatio-temporal predicates from spatial predicates (by lifting and aggregating). (Note that an arrowhead indicates which object's lifetime has to be considered in addition to the intersection of both objects' lifetimes.)

$$\begin{array}{rcl} \dot{p} & := & \exists p \\ \overleftrightarrow{p} & := & \forall_\cup p \\ \overleftarrow{p} & := & \forall_{\pi_1} p \\ \overrightarrow{p} & := & \forall_{\pi_2} p \\ \overline{p} & := & \forall_\cap p. \end{array}$$

There is a partial ordering among the universal quantifications. This is expressed by the following lemma:

Lemma 1 (Quantification Ordering).

1. $\overleftrightarrow{p}(S_1, S_2) \implies \overleftarrow{p}(S_1, S_2) \implies \overline{p}(S_1, S_2).$
2. $\overleftrightarrow{p}(S_1, S_2) \implies \overrightarrow{p}(S_1, S_2) \implies \overline{p}(S_1, S_2).$

Proof. Consider, for example, the first implication of 1(a).

$$\begin{array}{rcl} \overleftrightarrow{p}(S_1, S_2) & \iff & \forall_\cup p(S_1, S_2) \\ & \iff & \forall t \in dom(S_1) \cup dom(S_2) : p(S_1(t), S_2(t)) \\ & \implies & \forall t \in dom(S_1) : p(S_1(t), S_2(t)) \\ & \iff & \forall_{\pi_1} p(S_1, S_2) \\ & \iff & \overleftarrow{p}(S_1, S_2). \end{array}$$

The other implications are proved analogously. □

4.3 Basic Spatio-Temporal Predicates

Basic spatio-temporal predicates can be defined by temporal lifting and aggregation. We observe that, for each lifted spatial predicate, there is a "preferred" temporal aggregation. For example, when we ask by using a *disjoint* predicate whether the route of a plane did not encounter a storm, we usually require disjointedness only on the common lifetime, that is, the result of this query is not affected by the fact that either the storm or the flight started or ended before the respective other object. Thus, the preferred or default interpretation for spatio-temporal *disjoint* is the predicate $\overline{disjoint}$. In contrast, the query asking for species living in a certain climatic region usually implicitly asks for the species' lifetime being included in that of the climate region. This means, when we use spatio-temporal *inside*, we mean $\overleftarrow{inside}$.

Below, we give the definition for spatio-temporal predicates based on the default expected aggregation behavior. Usually, the expected behavior for $meet$ and $overlap$ is $\dot{meet}$ and $\dot{overlap}$, respectively. However, in the following, we are mainly working with universal quantifiers (to build complex predicates), and we have therefore chosen a $\forall$-definition for all predicates.

$$\begin{array}{rcl} Disjoint & := & \overline{disjoint} \\ Meet & := & \overleftrightarrow{meet} \\ Overlap & := & \overleftrightarrow{overlap} \\ Equal & := & \overleftrightarrow{equal} \\ Covers & := & \overrightarrow{covers} \\ CoveredBy & := & \overleftarrow{coveredBy} \\ Contains & := & \overrightarrow{contains} \\ Inside & := & \overleftarrow{inside}. \end{array}$$

The choice of defaults is to some degree debatable, and if, for instance, the symmetric definitions for *Meet* and *Overlap* are considered too restrictive, we can easily add further predicates with different aggregation behavior:

$$\begin{array}{rcl} Meets & := & \overleftarrow{meet} \\ MetBy & := & \overrightarrow{meet} \\ Overlaps & := & \overleftarrow{overlap} \\ OverlappedBy & := & \overrightarrow{overlap}. \end{array}$$

A further basic spatio-temporal predicate, which will be needed in the subsequent sections, is the predicate that yields true for two arbitrary spatio-temporal objects:

$$True \ := \ \lambda(S_1, S_2).T.$$

True can be well thought of as the universal quantification of the spatial predicate $true = \lambda(s_1, s_2).T$ (that yields true for any pair of spatial objects), that is, $True = \overleftrightarrow{true} = \ldots = true$.

Next, we consider how to combine basic spatio-temporal predicates into complex ones.

4.4 Developments: Sequences of Spatio-Temporal Predicates

In this section, we will demonstrate that particular relationships between two spatio-temporal objects can be appropriately modeled by sequences of spatial and (basic) spatio-temporal predicates.

Consider, for example, the following scenario: A moving point P is located at time t_1 outside of an evolving region R and changes (continuously) its location so that at time t_3 it is inside of R. (We assume in the following that $i < j \implies t_i < t_j$.) This could be paraphrased as "P *enters* R." If P, at some time t_5, is again outside of R, we could call the whole story (that is, the development of P with respect to R during the time interval $[t_1, t_5]$) "P *crosses* R."

In these two examples, we notice that the continuous movement of P implies that P is located on the border of R at some time t_2 and at some time t_4. If at all other times P is located either inside or outside of R (that is, P does not move along/stay on the border of R), we can characterize the development of P more precisely as follows:

predicate	holds	
$Disjoint(P, R)$	during	$[t_1, t_2[$
$meet(P, R)$	at	t_2
$Inside(P, R)$	during	$]t_2, t_4[$
$meet(P, R)$	at	t_4
$Disjoint(P, R)$	during	$]t_4, t_5]$

We can observe two things: First, in order to specify developments of objects, we have to restrict the validity of spatio-temporal predicates to intervals and, second, some predicates, such as *disjoint* and *inside*, are used for whole time intervals whereas other predicates, such as *meet*, are used only at time points.

The first observation leads us to the following definition (where $S|_I$ denotes the partial function that yields $S(t)$ for $t \in I$ and is undefined otherwise):

Definition 2 (Predicate Constriction). *Let P be a spatio-temporal predicate, and let I be a (half-) open or closed interval. Then,*

$$P_I := \lambda(S_1, S_2).P(S_1|_I, S_2|_I).$$

For an example, consider P and R from above. It is clear that $Inside(P, R)$ is false, but $Inside|_{]t_2,t_4[}(P, R)$ is true. In the following, we will abbreviate constrictions of P with open intervals $]t, \infty[$ and $]-\infty, t[$ by simply writing $P_{>t}$ and $P_{<t}$, respectively.

From the definition of constriction and from the definition of function restriction, it follows that successive constrictions can be accumulated by intersecting the corresponding constriction intervals:

Lemma 2 (Constriction Composition). $(P|_I)|_{I'} = P|_{I \cap I'}$.

Proof.

$$\begin{aligned}(P|_I)|_{I'}(S_1, S_2) &= P|_I(S_1|_{I'}, S_2|_{I'}) \\ &= P((S_1|_{I'})|_I, (S_2|_{I'})|_I) \\ &= P(S_1|_{I' \cap I}, S_2|_{I' \cap I}) \\ &= P|_{I' \cap I}(S_1, S_2).\end{aligned}$$

□

The interesting aspect of the second observation is that some predicates actually *cannot* hold at time points. For example, it is generally not possible for *disjoint* and *inside* to hold only at one time point when describing relationships of spatial objects over some time interval.[1] On the other hand, any predicate can principally hold for a period of time. For example, *meet* holds for a whole interval if the point moves along the border of R or stays stationary on the border for some time.

This means, we can actually identify two classes of predicates: 1) predicates that can be true for an instant in time and 2) predicates that can, in general, only hold for a period of time. Accordingly, we call the former ones *instant predicates* and the latter ones *period predicates*. The classification is given in the following table:

instant predicates	period predicates
equal	*disjoint*
meet	*overlap*
covers, *coveredBy*	*inside*, *contains*

1. It is, however, possible that, say, *disjoint*, holds only at a time point t if one of the objects exists only at t (that is, is undefined for $t' \in [t-\epsilon, t+\epsilon] - \{t\}, \epsilon > 0$) or if one of the objects makes a completely discontinuous movement.

Recall that, while period predicates cannot hold at isolated time points, instant predicates can easily be used at periods. We will need this distinction of predicates later when we define the semantics of sequences of predicates.

A further observation with regard to instant/period predicates is that only those developments can be satisfied in which two different period predicates are "connected" by an instant predicate. For instance, it is not possible for two spatio-temporal objects to satisfy *Inside* immediately after *Disjoint* (under the assumption of continuous movement).

Next, we define three operations for combining spatio-temporal and spatial predicates: $p \vdash P$ (*from*) defines a spatio-temporal predicate that for some time t_0 checks p and then enforces P for all $t > t_0$; $P \dashv p$ (*until*) is defined dually, that is, P must hold until p is true at some time t_0. Finally, $P \dashv p \vdash Q$ (*then*) is true if there is some time point t_0 when p is true so that P holds before and Q holds after t_0.

Definition 3 (Temporal Composition). *Let p be a spatial predicate, and let P and Q be spatio-temporal predicates. Then,*

$$\begin{aligned}p \vdash P &:= \lambda(S_1, S_2).\exists t : p(S_1(t), S_2(t)) \wedge P_{>t}(S_1, S_2) \\ P \dashv p &:= \lambda(S_1, S_2).\exists t : p(S_1(t), S_2(t)) \wedge P_{<t}(S_1, S_2) \\ P \dashv p \vdash Q &:= \lambda(S_1, S_2).\exists t : p(S_1(t), S_2(t)) \\ &\qquad \wedge P_{<t}(S_1, S_2) \wedge Q_{>t}(S_1, S_2).\end{aligned}$$

These combinators are related in several ways. First, we know:

Lemma 3. *If $P \dashv p \vdash Q$ holds for two spatio-temporal objects S_1 and S_2, then so does $P \dashv p$ and also $p \vdash Q$.*

Proof. Consider two spatio-temporal objects S_1 and S_2 for which $P \dashv p \vdash Q$ holds. By definition of $P \dashv p \vdash Q$, there must be a time point t such that 1) $p(S_1(t), S_2(t))$, 2) $P_{<t}(S_1, S_2)$, and 3) $Q_{>t}(S_1, S_2)$ are true. From 1 and 2, it follows that $P \dashv p$ holds for S_1 and S_2 and, from 1 and 3, it follows that $p \vdash Q$ holds for S_1 and S_2. □

It is important to note that the other direction does not hold, that is, even if both predicates $P \dashv p$ and $p \vdash Q$ are true for two objects, this does not necessarily mean that $P \dashv p \vdash Q$ is also true. This is because the t values that make $P \dashv p$ and $p \vdash Q$ true can be chosen independently from one another, whereas we need exactly one t value to make $P \dashv p \vdash Q$ true. This means that the *then* predicate is really needed and cannot be expressed by a simple conjunction of *from* and *until* (we rather have to "synchronize" on the t value). On the other hand, *from* and *until* are special cases of *then*. This follows from the fact that *True* is a unit for temporal composition:

Lemma 4 (Derived Compositions).

1. $p \vdash P = True \dashv p \vdash P$.
2. $P \dashv p = P \dashv p \vdash True$.

Proof. Consider, for example, item 1. $p \vdash P$ yields true for two objects S and S' if there is a t such that $p(S(t), S'(t))$ and $P_{>t}(S, S')$ is true. On the other hand, $True \dashv p \vdash P$ yields true for S and S' if $p(S(t), S'(t))$, $P_{>t}(S, S')$, and $True_{<t}(S, S')$ are true, which is identical to the previous condition since

$$True_{<t}(S, S') = True(S_{<t}, S'_{<t}) = T.$$

Item 2 is proved in a similar way. □

Hence, we can always restrict ourselves to *then* compositions knowing that all results extend to *from* and *until* by the expansion described by Lemma 4.

4.5 A Concise Syntax for Developments

We can exploit the fact that temporal composition is associative to define a concise sequencing syntax for predicates.

Lemma 5 (Associativity of Composition).

$$P \dashv p \vdash (Q \dashv q \vdash R) = (P \dashv p \vdash Q) \dashv q \vdash R.$$

Proof. We have to show that whenever the left-hand side is true for two objects S and S', then so is the right-hand side, and vice versa. If the left-hand side is true, we know that there is a time point t_1, such that $p(S(t_1), S'(t_1))$, $P_{<t_1}(S, S')$, and $(Q \dashv q \vdash R)_{>t_1}(S, S')$ is true. The last condition can be reformulated as follows:

$$\begin{aligned}&(Q \dashv q \vdash R)_{>t_1}(S, S')\\ &= (Q \dashv q \vdash R)(S_{>t_1}, S'_{>t_1})\\ &= \exists t_2 : q(S_{>t_1}(t_2), S'_{>t_1}(t_2)) \wedge Q_{<t_2}(S_{>t_1}, S'_{>t_1})\\ &\quad \wedge R_{>t_2}(S_{>t_1}, S'_{>t_1}).\end{aligned}$$

In the last line, the first condition can be simplified further by observing that q can only be true on defined values, and since $S_{>t_1}$ and $S'_{>t_1}$ return defined values only for time points greater than t_1, q can be true only for time points $t_2 > t_1$. Moreover, the second and third conditions can be simplified by applying Lemma 2, that is,

$$Q_{<t_2}(S_{>t_1}, S'_{>t_1}) = Q|_{]t_1,t_2[}(S, S')$$

and

$$R_{>t_2}(S_{>t_1}, S'_{>t_1}) = R_{>t_2}(S, S').$$

The latter equation is due to the fact that

$$t_2 > t_1 \implies]t_1, \infty[\cap]t_2, \infty[=]t_2, \infty[.$$

Next, the condition $p(S(t_1), S'(t_1))$ is equivalent to $p(S_{<t_2}(t_1), S'_{<t_2}(t_1))$ since $t_1 < t_2$, and $P_{<t_1}(S, S')$ implies that $P_{<t_1}(S_{<t_2}, S'_{<t_2})$ is true. Finally, $Q|_{]t_1,t_2[}(S, S')$ can be also written as $Q_{>t_1}(S_{<t_2}, S'_{<t_2})$. Thus, we can derive:

$$\begin{aligned}&\exists t_1 : p(S_{<t_2}(t_1), S'_{<t_2}(t_1)) \wedge P_{<t_1}(S_{<t_2}, S'_{<t_2})\\ &\quad \wedge Q_{>t_1}(S_{<t_2}, S'_{<t_2})\\ &= (P \dashv p \vdash Q)(S_{<t_2}, S'_{<t_2})\\ &= (P \dashv p \vdash Q)_{<t_2}(S, S').\end{aligned}$$

The last condition together with the conditions $q(S(t_2), S'(t_2))$ and $R_{>t_2}(S, S')$ show that the right-hand side also holds for S and S'. The other direction is proved analogously. □

In order to be able to denote developments concisely, we use some syntactic sugar in writing down cascades of compositions: Compositions are denoted simply by a sequence of predicates linked together by an infix-operator $\triangleright$. More precisely, we allow only alternating sequences of spatial and spatio-temporal predicates of length ≥ 2. This means, the language Π of predicate sequences is given by the following regular expression (we use the abbreviation X^c that denotes the nonempty sequences of Xs separated by cs)

$$\Pi = (p \triangleright P)^{\triangleright}[\ \triangleright p] \mid (P \triangleright p)^{\triangleright}[\ \triangleright P].$$

The translation into (nested) compositions is done by the mapping $\mathcal{C}$ defined below. Note that the first case is needed to capture some recursive calls of $\mathcal{C}$ (for example, in the translation of $P \triangleright p \triangleright Q$)

$$\begin{aligned}\mathcal{C}(P) &= P\\ \mathcal{C}(P \triangleright p) &= P \dashv p\\ \mathcal{C}(P \triangleright p \triangleright \Pi) &= P \dashv p \vdash \mathcal{C}(\Pi)\\ \mathcal{C}(p \triangleright P) &= p \vdash P\\ \mathcal{C}(p \triangleright P \triangleright \Pi) &= p \vdash \mathcal{C}(P \triangleright \Pi).\end{aligned}$$

Now, by the term "development," we specifically mean a sequence of predicates combined by $\triangleright$. As an example, consider the development

$$Disjoint \triangleright meet \triangleright Inside \triangleright meet \triangleright Disjoint.$$

This is an abbreviation for (that is, is translated by $\mathcal{C}$ into)

$$Disjoint \dashv meet \vdash (Inside \dashv meet \vdash Disjoint).$$

(Since composition is an associative operation, the chosen nesting does not matter.)

We have defined $\triangleright$ so that only alternating sequences of spatio-temporal and spatial predicates can be built. However, sometimes it is convenient to be able to omit a spatial predicate next to its corresponding instant spatio-temporal predicate, for example, we would like to write $Disjoint \triangleright Meet$ instead of $Disjoint \triangleright meet \triangleright Meet$. Therefore, we allow a corresponding shortcut in our notation which is justified by the fact that whenever an instant spatio-temporal predicate holds immediately before or after another spatio-temporal predicate, then there are precisely defined time points where the predicate holds the first and the last time. Let i be an instant spatial predicate, and let I be its corresponding spatio-temporal predicate. Here, P and Q denote either basic spatio-temporal predicates or predicate sequences that end (in the case of P) or start (in the case of Q) with a basic spatio-temporal predicate. Then, we allow the following abbreviations:

Abbreviation	Expands to
$I \triangleright Q$	$I \triangleright i \triangleright Q$
$P \triangleright I$	$P \triangleright i \triangleright I$
$P \triangleright I \triangleright Q$	$P \triangleright i \triangleright I \triangleright i \triangleright Q$

Finally, we have to describe how existentially quantified predicates fit into the sequencing syntax. In general, we would like to be able to put existentially quantified predicates next to other spatio-temporal predicates *without* having to give a connecting spatial predicate. For example, we would like to express the fact that two disjoint objects meet at some time later simply by writing

$$Disjoint \triangleright meet.$$

Fortunately, we can always expand an existentially quantified predicate into a sequence containing only spatial and universally quantified predicates. This is captured in the following result:

Lemma 6 (Existential Expansion). $\exists p = True \triangleright p \triangleright True.$

Proof.

$$\begin{aligned} True \triangleright p \triangleright True &= True \dashv p \vdash True \\ &= \lambda(S_1, S_2).\exists t : p(S_1(t), S_2(t)) \\ &\quad \wedge True_{<t}(S_1, S_2) \wedge True_{>t}(S_1, S_2) \\ &= \lambda(S_1, S_2).\exists t : p(S_1(t), S_2(t)) \\ &= \exists p. \end{aligned}$$

□

Thus, the above example is expanded to

$$Disjoint \triangleright True \triangleright meet \triangleright True$$

which is then further expanded to

$$Disjoint \triangleright true \triangleright True \triangleright meet \triangleright True$$

(which can also be transformed into a nested *then* expression).

4.6 Advanced Predicate Combinators

Beyond the sequencing of basic spatio-temporal predicates, there are several other logical connectives that can be defined to combine predicates.

One example is the ability to express disjunctions of spatio-temporal predicates.

Definition 4 (Temporal Alternative). *Let P and Q be two spatio-temporal predicates. The temporal alternative between P and Q is defined as:*

$$P \mid Q := \lambda(S_1, S_2).P(S_1, S_2) \vee Q(S_1, S_2).$$

Consider, for example, a moving point on the border of a region. The situations that can arise when the point leaves the border are captured by the alternative:

$$Disjoint \mid Inside.$$

It remains to be explained what the meaning of alternatives within developments is. For example, when we have to describe the situation that a point eventually leaves the border of a region, we would like to be able to simply write (note that we give $\mid$ a higher precedence than $\triangleright$):

$$Meet \triangleright Disjoint \mid Inside$$

(which is expanded to $Meet \triangleright meet \triangleright Disjoint \mid Inside$.) Fortunately, $\triangleright$ distributes over $\mid$ from the left as well as from the right. This means:

Lemma 7 (Distributivity of Composition).

1. $P \triangleright p \triangleright (Q \mid R) = (P \triangleright p \triangleright Q) \mid (P \triangleright p \triangleright R).$
2. $(P \mid Q) \triangleright p \triangleright R = (P \triangleright p \triangleright R) \mid (Q \triangleright p \triangleright R).$

Proof. Consider item 1. $P \triangleright p \triangleright Q \mid R$ is true for two spatio-temporal objects S and S' if there is some time point t, so that

1. $p(S(t), S'(t))$,
2. $P_{<t}(S, S')$, and
3. $(Q \mid R)_{>t}(S, S')$

are true. Unrolling the definitions of constriction and alternative, the latter condition is equivalent to $Q(S_{>t}, S'_{>t}) \vee R(S_{>t}, S'_{>t})$. Since $\wedge$ distributes over $\vee$, we can rewrite the whole condition as a disjunction:

$$((1) \wedge (2) \wedge Q(S_{>t}, S'_{>t})) \vee ((1) \wedge (2) \wedge R(S_{>t}, S'_{>t})).$$

The first alternative characterizes the development $P \triangleright p \triangleright Q$, and the second one describes $P \triangleright p \triangleright R$. By folding the definition of temporal alternative, we obtain the right-hand side. Item 2 is proved similarly. □

This means that we can always convert developments containing alternatives into what we call *development normal form*, that is, an alternative of developments each of which does not contain any alternatives; for these the above syntax rules apply. Note, however, that $\mid$ does not distribute over $\triangleright$. To investigate this topic, we first have to syntactically fix what distribution over a composition means at all. Since temporal alternative is only defined for spatio-temporal predicates, a reasonable view is that the alternative is moved to all spatio-temporal predicates in the composition. Thus, in general, we have:

$$P \mid (Q \triangleright p \triangleright R) \neq (P \mid Q) \triangleright p \triangleright (P \mid R).$$

This can be seen as follows: Let

$$P = Disjoint, Q = R = Inside,$$

and $p = meet$. Now, a moving point and an evolving region that are disjoint certainly fit the predicate on the left-hand side (because of the alternative P), but they fail for the right-hand side because there it is required that they meet.

A further operation on spatio-temporal predicates is negation which is defined as follows:

Definition 5 (Predicate Negation). *Let P be a spatio-temporal predicate. The temporal negation of P is defined as:*

$$\sim P := \lambda(S_1, S_2).\neg(P(S_1, S_2)).$$

Next, we define a "backward" or "reverse" combinator for spatio-temporal predicates. The definition is based on a notion of "reflection" for spatio-temporal objects. This means we define that the reverse of a development P is true for two objects if and only if P holds for the reflection of these objects.

The reflection of a single object S can be explained as follows: Imagine S exists in an interval $[t_1, t_2]$ (that is, $dom(S) = [t_1, t_2]$), and S moves (and possibly changes its shape) from its initial value $S(t_1)$ to $S(t_2)$. Then, the corresponding reflected object $reflect(S)$ is given by moving backward in the same interval from $S(t_2)$ to $S(t_1)$. Formally, we can achieve this effect by defining $reflect(S)$ to yield at time t the value that S had at time

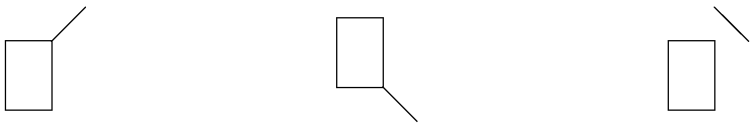

Fig. 2. Object reflection.

$$\sup(dom(S)) - t + \inf(dom(S)).$$

Here, $\sup$ and $\inf$ denote the supremum and infimum of a set; their use is required in those cases when the domain of a spatio-temporal object is a (half-) open set.

Now, since predicates are defined on pairs of objects, we need a notion of reflection for two objects. The definition is similar to that for a single object, but we have to take into account the union of both objects' lifetimes. This means that both objects are not reflected independently of one another but are possibly shifted in time toward an earlier beginning or a later ending of their companion object. (This will be illustrated in a picture below.)

Therefore, we define a function *reflect* that takes two spatio-temporal objects and returns the pair of reflected objects.

Definition 6 (Object Reflection). *Let S_1 and S_2 be two spatio-temporal objects. The reflection of S_1 and S_2 is defined as:*

$$\begin{aligned} reflect(S_1, S_2) := (&\lambda t \in D.S_1(t_s - t + t_i),\\ &\lambda t \in D.S_2(t_s - t + t_i))\\ &\text{where } D = dom(S_1) \cup dom(S_2)\\ &t_s = sup(D)\\ &t_i = \inf(D). \end{aligned}$$

Note again that the above definition, which reflects both objects with respect to the supremum of both objects' lifetimes, is different than reflecting both objects in isolation and taking the pair of these reflections. To see this, consider a (constant, nonmoving) region R existing during an interval $[t_1, t_2]$, and a moving point P existing during the interval $[t_2, t_3]$ that touches R and then moves away. This is shown in the left part of Fig. 2 (where we show a 2D projection and assume that time increases from bottom to top). Now, $reflect(R, P)$ yields the objects shown in the middle, in contrast to the pair of isolated reflected objects that are shown on the right.

Now, we can define the backward combinator based on object reflection. The definition below expresses that the reverse of P holds for two objects S_1 and S_2 if P holds for the reflection of S_1 and S_2. This is much simpler than to define reflection inductively on the construction of predicates.

Definition 7 (Predicate Reflection). *Let P be a spatio-temporal predicate. The reflection of P is defined as:*

$$P^{\leftarrow} := \lambda(S_1, S_2).P(reflect(S_1, S_2)).$$

Predicate reflection obeys a number of interesting laws. For example, reflection does not change basic spatio-temporal predicates, or reflection reverses the order of predicates in a composition, see Section 4.9.

Finally, we introduce the following predicate combinators that prove useful in the specification of spatio-temporal predicates: P^+ holds if P holds repeatedly, but at least once, that is, a development given by a nonempty sequence of P's holds. P^* is similar to P^+ except that P need not hold at all. $P \,\&\, Q$ denotes spatio-temporal conjunction of predicates. $P \,\&\, Q$ holds whenever P and Q hold for a pair of objects. Likewise, $P \rightarrow Q$ denotes spatio-temporal implication.

Definition 7 (Derived Combinators). *Let P and Q be arbitrary spatio-temporal predicates. Then:*

$$\begin{aligned} P^+ &:= P \triangleright P^*\\ P^* &:= True \mid P^+\\ P \,\&\, Q &:= \sim(\sim P \mid\sim Q)\\ P \rightarrow Q &:= \sim P \mid Q. \end{aligned}$$

Here, the definitions for & and $\rightarrow$ are based on the combinators for temporal alternative (Definition 4) and predicate negation (Definition 5). This is a bit shorter than giving the object-based definitions using logical $\wedge$ and $\Rightarrow$ and, more importantly, it simplifies proofs involving these operators, for example, Lemma 10. It is easy to see that the object-based definitions using logical operators are equivalent. For instance, for &, we could give the following definition:

$$P \,\&\, Q := \lambda(S_1, S_2).P(S_1, S_2) \wedge Q(S_1, S_2)$$

and we can show:

$$\begin{aligned} &\lambda(S_1, S_2).P(S_1, S_2) \wedge Q(S_1, S_2)\\ &\quad = \lambda(S_1, S_2).\neg(\neg P(S_1, S_2) \vee \neg Q(S_1, S_2))\\ &\quad = \sim(\lambda(S_1, S_2).\neg P(S_1, S_2) \vee \neg Q(S_1, S_2))\\ &\quad = \sim(\lambda(S_1, S_2).\neg P(S_1, S_2) \mid \lambda(S_1, S_2).\neg Q(S_1, S_2))\\ &\quad = \sim(\sim P \mid\sim Q). \end{aligned}$$

Examples of the newly introduced combinators will be presented in Section 4.8.

4.7 Unary Predicates and Sectioning

Until now we have assumed that spatio-temporal predicates are always binary. However, it is obvious that there are useful predicates of arbitrary arity. In this section, we focus on the definition and use of unary predicates; in particular, we will introduce the concept of "sectioning" of binary predicates to obtain unary predicates. We will not deal with predicates of arity ≥ 3; these could be made easily available by introducing lambda-abstraction as a construct in the specification language.

For each binary predicate or function, we can consider its *left* or *right section* which is the predicate or function with its left or right parameter fixed, and the process of building such a section is called *sectioning*. Sectioning is a popular syntactic tool in functional programming that allows us to

denote certain functions in a highly concise way (see [4]). For example, $(1+)$ fixes the first argument of the addition function with 1 and, thus, denotes the successor function. Likewise, (-1) denotes the predecessor function (note that this is different from $(1-)$), $(*2)$ denotes the function that doubles its argument, and (< 3) is the predicate that yields true for all numbers that are smaller than 3. For spatio-temporal predicates, we can define sectioning as follows:

Definition 9 (Sectioning). *Let P be a spatio-temporal predicate, and let S be a spatio-temporal object.*

$$(S\ P) := \lambda S'.P(S, S') \quad (Left\ Section)$$
$$(P\ S) := \lambda S'.P(S', S) \quad (Right\ Section).$$

As a simple example, assume R is an evolving region. Now, consider the right section

$$(Inside\ R).$$

This denotes a unary spatio-temporal predicate that checks for a spatio-temporal object whether it is always inside R or not. In contrast, the left section

$$(S\ Inside)$$

is a unary predicate that checks for an evolving region whether it contains S or not (where S might be either an evolving region or a moving point).

Sectioning is especially useful for expressing relationships between more than two objects. For example, consider animals that cross a lake (L) and after that disappear in the woods (W). These animals satisfy the predicate:

$$(Cross\ L) \triangleright True \triangleright (Enter\ W).$$

The reader might have noticed that the above expression contains, strictly speaking, a type error because we are using composition (which has been defined only for binary predicates) on unary predicates. For simplicity, we have deliberately given in Definition 3 only a specific instance, namely, on two arguments, of the more general version. However, composition can be defined in a completely generic way as a function of type

$$(\tau(\alpha) \to Bool) \times (\alpha \to Bool) \times (\tau(\alpha) \to Bool) \to (\tau(\alpha) \to Bool)$$

as follows:

$$P \dashv p \vdash Q := \lambda x.\exists t : p(x(t)) \wedge P_{<t}(x) \wedge Q_{>t}(x).$$

Now, Definition 3, that works for pairs, is an instance of the above definition with the type $\beta_1 \times \beta_2$ substituted for α (because of the isomorphism $\tau(\beta_1 \times \beta_2) \cong \tau(\beta_1) \times \tau(\beta_2)$). (Similarly, we can imagine $True$ as being defined in a generic way for an arbitrary number of arguments.)

4.8 Examples

Next, we present some examples. The above mentioned predicates "enter" and "cross" (for a point and a region) can now be simply defined by:

$$Enter := Disjoint \triangleright meet \triangleright Inside$$
$$Cross := Disjoint \triangleright meet \triangleright Inside \triangleright meet \triangleright Disjoint.$$

It seems that predicate specifications become longer very quickly—consider, for example, the definition of Cross for two evolving regions:

$$Cross := Disjoint \triangleright meet \triangleright Overlap \triangleright coveredBy \triangleright Inside \triangleright coveredBy \triangleright Overlap \triangleright meet \triangleright Disjoint.$$

However, since sequential composition is associative, we can easily define complex predicates in a modular way by successively composing simpler predicates. For example, we can define *Cross* by composing *Enter* and a corresponding predicate *Leave*:

$$Leave := Enter^{\leftarrow}$$
$$Cross := Enter \triangleright Leave.$$

To see that this definition is equivalent to the previous one, we have to prove that $Inside \triangleright Inside = Inside$ and that $Enter^{\leftarrow} = Inside \triangleright meet \triangleright Disjoint$. We shall consider such laws in the next section.

A note on macros and syntax: The expansion of macro definitions, such as *Enter* or *Leave* in *Cross*, is performed before any other syntactic rule is applied.

By using $\forall$-predicates, developments of objects are specified very precisely. In contrast, more relaxed specifications can be obtained by exists predicates. For example, an alternative specification of *Cross* that does not care about the precise process of entering and leaving can be simply given by:

$$Cross := Disjoint \triangleright inside \triangleright Disjoint.$$

The right-hand side expands to

$$Disjoint \triangleright true \triangleright True \triangleright inside \triangleright True \triangleright true \triangleright Disjoint,$$

which nicely demonstrates how $True$ works as a wildcard in development specifications. An interesting observation is that, whereas the first definition for $Cross$ works only in the point/region case, the latter definition also works in the region/region case.

A point object that is located on the border of a region and that might temporarily leave the border an arbitrary number of times can be specified by:

$$TempLeave := (Meet \triangleright Disjoint)^* \triangleright Meet.$$

Examples for & are difficult to find since, for example, two distinct basic spatio-temporal predicates cannot be satisfied at the same time—only in sequence. However, to specify subsequent predicates, we have sequencing already available. In fact, $\triangleright$ plays the role of a "multiplication operation" with respect to spatio-temporal predicates (see next section), and & is just included for completeness.

Finally, as an example for spatio-temporal implication consider the predicate

$$Visitor := (Enter \triangleright True) \to (True \triangleright Leave).$$

This predicate yields true for objects that always leave whenever they enter a region. In particular, the predicate is true for objects that are always inside or disjoint from the evolving region. Hence, it is not identical to $Cross$, but

rather subsumes it as a special case. An application is traveling: A foreigner that never enters a country does not have to leave it; likewise, inhabitants of the country do not enter it, but may leave it, only when a foreigner enters does she later have to leave the country again.

4.9 An Algebra of Spatio-Temporal Predicates

What we have defined so far is a kind of algebra with spatio-temporal predicates as objects and combinators, such as $\triangleright$ and $\mid$, as operations. More precisely, $\mid$ is a kind of summation operation, whereas $\triangleright$ plays the role of multiplication. In this section, we present a collection of useful laws for this algebra.

First, we can look at some very elementary relationships between different operators and predicates. For example, $True$ is a unit with respect to $\mid$ and is a left (right) unit with respect to developments that begin (end) with a spatial predicate. Moreover, $True$ is not affected by constriction. Likewise, $False$ (which is defined as $\lambda(S_1, S_2).F$) is a zero with respect to $\mid$ and $\triangleright$. Like $True$, $False$ is not affected by constriction.

Lemma 8 (Zero & Unit).

1. $True \mid P = True = P \mid True.$
2. $False \mid P = P = P \mid False.$
3. $True \triangleright p \triangleright P = p \triangleright P.$
4. $P \triangleright p \triangleright True = P \triangleright p.$
5. $False \triangleright p \triangleright P = False = p \triangleright P \triangleright False.$
6. $True|_I = True.$
7. $False|_I = False.$

Proof. Cases 1 and 2 follow directly from Definition 4 since an alternative of spatio-temporal predicates is reduced to $\vee$ and since $True(False)$ yields always $T(F)$ for two objects.

Cases 3 and 4 just rephrase Lemma 4 in sequencing syntax. Case 5 follows immediately from Definition 3 since, for example, $False \triangleright p \triangleright P$ requires the existence of a time point t so that $False_{<t}$ is true. However, this is not possible (see also 7).

The cases 6 and 7 hold because $True$ $(False)$ yields T (F) irrespective of the domains of the spatio-temporal objects, which are the only parts affected by constriction. □

We have already seen in Lemma 5 that predicate composition is associative. The same is true for disjunction. Moreover, disjunction is also commutative.

Lemma 9 (Associativity and Commutativity of Alternative).

1. $P \mid (Q \mid R) = (P \mid Q) \mid R.$
2. $P \mid Q = Q \mid P.$

Proof. Both results follow directly from Definition 4 together with the fact that the corresponding property holds for logical or. □

An immediate consequence of the definition of negations is:

Corollary 1 (Double Negation). $\sim (\sim (P)) = P.$

We also have a correspondence of de Morgan's laws:

Lemma 10 (De Morgan's Laws).

1. $\sim (P \mid Q) = \sim P \,\&\, \sim Q.$
2. $\sim (P \,\&\, Q) = \sim P \mid\sim Q.$

Proof. For case 1, we have: $\sim P \,\&\, \sim Q = \sim (\sim (\sim P) \mid\sim (\sim Q))$ by Definition 8. By Corollary 1, this is equal to $\sim (P \mid\sim Q)$.

In case 2, we know from Definition 8 that $\sim (P\& Q) = \sim (\sim (\sim P \mid\sim Q))$. By Corollary 1, this is equal to $\sim P \mid\sim Q$. □

Predicate reflection abounds in useful laws. Some are given in the following lemma.

Lemma 11 (Reflection Propagation).

1. $P^{\leftarrow} = P$ *if* P *is a basic spatio-temporal predicate.*
2. $(P^{\leftarrow})^{\leftarrow} = P.$
3. $(P \triangleright p \triangleright Q)^{\leftarrow} = Q^{\leftarrow} \triangleright p \triangleright P^{\leftarrow}.$

Proof. Part 1 follows directly from the definition of existential and universal quantification because a systematic temporal reordering of object values as performed by *reflect* does not affect the validity of predicates and the temporal aggregation either.

Next, consider part 2. First, we can insert the definition of predicate reflection twice and obtain:

$$\begin{aligned}(P^{\leftarrow})^{\leftarrow}(S_1, S_2) &= (P^{\leftarrow})(reflect(S_1, S_2)) \\ &= P(reflect(reflect(S_1, S_2))).\end{aligned}$$

Next, we need the property that object reflection is self-inverse. In the following, let D, t_s, and t_i be defined as in Definition 6. We show the effect of double reflection on the first argument (S_1); the case for S_2 is analogous.

$$\begin{aligned}&reflect(reflect(S_1, S_2)) \\ &\quad = reflect(\lambda t \in D.S_1(t_s - t + t_i), \ldots) \\ &\quad = (\lambda t' \in D'.(\lambda t \in D.S_1(t_s - t + t_i))\ (t'_s - t' + t'_i), \ldots) \\ &\quad = (\lambda t' \in D'.S_1(t_s - (t'_s - t' + t'_i) + t_i), \ldots) \\ &\quad = (\lambda t' \in D'.S_1(t_s - t'_s + t' - t'_i + t_i), \ldots).\end{aligned}$$

Now, we observe that the union of the domains of S_1 and S_2 is equal to the union of the domains of the corresponding reflected objects. This means that $D' = D$ and, in particular, that $t'_s = t_s$ and $t'_i = t_i$. Hence, the above expression can be simplified to

$$(\lambda t' \in D'.S_1(t'), \ldots)$$

which is equal to $(S_1, \ldots)$. Performing the same transformation for the second argument thus yields that $reflect \circ reflect$ is the identity function on pairs of spatio-temporal objects. This shows that $P(reflect(reflect(S_1, S_2))) = P(S_1, S_2)$ which completes the proof of 2.

Now, consider part 3. We begin by unrolling the definition of reflection:

$$(P \triangleright p \triangleright Q)^{\leftarrow}(S_1, S_2) = (P \triangleright p \triangleright Q)(reflect(S_1, S_2)).$$

Let $(S'_1, S'_2) = reflect(S_1, S_2)$. Now, the right-hand side is true if there is a time point t such that 1) $p(S'_1(t), S'_2(t))$, 2) $P_{<t}(S'_1, S'_2)$, and 3) $Q_{>t}(S'_1, S'_2)$ are true. By the definition of *reflect*, t was mapped from some time point

t'. Thus, p holds for the original, unreflected object pair (S_1, S_2) at t', that is,

$$p(S_1(t'), S_2(t')). \quad (*)$$

Again by the definition of *reflect*, all time points $< t$ are mapped from all those time points $> t'$, and all time points $> t$ are mapped from all those time points $< t'$. Hence, we also know that $P_{>t'}(S_1', S_2')$ and $Q_{<t'}(S_1', S_2')$ is true. This is equivalent to $P^{\leftarrow}_{>t'}(S_1, S_2)$ and $Q^{\leftarrow}_{<t'}(S_1, S_2)$, which together with $(*)$ is nothing but the definition for $Q^{\leftarrow} \triangleright p \triangleright P^{\leftarrow}$. □

There are several relationships that can be exploited for optimization or spatio-temporal reasoning in general. Imagine, for example, that predicates are used in a constraint database to enforce certain developments of objects which might often be done by listing alternative developments. Now, as a corollary of Lemma 7, we have:

Corollary 2 (Development Factorization).

$$(P \triangleright p \triangleright Q \triangleright q \triangleright R) \mid (P \triangleright p \triangleright Q' \triangleright q \triangleright R) = P \triangleright p \triangleright Q \mid Q' \triangleright q \triangleright R.$$

Proof. We can group subdevelopments explicitly. This means, we write the left-hand side as:

$$(P \triangleright p \triangleright (Q \triangleright q \triangleright R)) \mid (P \triangleright p \triangleright (Q' \triangleright q \triangleright R)).$$

Then, we can apply Lemma 7, item 1, from right to left and get

$$P \triangleright p \triangleright (Q \triangleright q \triangleright R) \mid (Q' \triangleright q \triangleright R).$$

Now, we can apply Lemma 7, item 2, to the alternative within the development and obtain

$$P \triangleright p \triangleright (Q \mid Q' \triangleright q \triangleright R).$$

Finally, ungrouping yields the right-hand side of the corollary. □

This result can be used to "factorize" common parts of alternative developments to simplify the enforcement of spatio-temporal constraints.

Our final result shows an example of how facts about spatial predicates can be lifted to spatio-temporal laws (in this case, even irrespective of the concrete $\forall$-semantics):

Theorem 1 (Theorem Lifting).

$$\neg p = q \Rightarrow \forall_\gamma q \rightarrow \sim (\forall_\gamma p) = True.$$

Proof. In the derivation below, we abbreviate

$$\gamma(dom(S_1), dom(S_2))$$

by Γ. We transform the LHS of the conclusion by using Definitions 8, 4, and 5, and by applying the $\forall$-definition.

$$\begin{aligned}
\forall_\gamma q \rightarrow \sim (\forall_\gamma p) &= \sim (\forall_\gamma q) \mid\sim (\forall_\gamma p) \\
&= \lambda(S_1, S_2). \sim (\forall_\gamma q)(S_1, S_2) \vee \sim (\forall_\gamma p)(S_1, S_2) \\
&= \lambda(S_1, S_2).\neg(\forall_\gamma q(S_1, S_2)) \vee \neg(\forall_\gamma p(S_1, S_2)) \\
&= \lambda(S_1, S_2).\neg(\forall t \in \Gamma : q(S_1(t), S_2(t))) \\
&\quad \vee \neg(\forall t \in \Gamma : p(S_1(t), S_2(t))) \\
&= \lambda(S_1, S_2).\exists t \in \Gamma : \neg q(S_1(t), S_2(t)) \\
&\quad \vee \exists t \in \Gamma : \neg p(S_1(t), S_2(t)).
\end{aligned}$$

Next, we can apply the precondition $\neg p = q$ to the second alternative and obtain

$$\begin{aligned}
&= \lambda(S_1, S_2).\exists t \in \Gamma : \neg q(S_1(t), S_2(t)) \vee \\
&\qquad \exists t \in \Gamma : q(S_1(t), S_2(t)).
\end{aligned}$$

Now, the body of the predicate has evolved into a tautology, that is, there must either be a $t \in \Gamma$ for which q is not true, or for all $t \in \Gamma$, we have q is true, which clearly implies that q is true for some time. Then, we continue

$$\begin{aligned}
&= \lambda(S_1, S_2).true \\
&= True.
\end{aligned}$$

Thus, we have shown the validity of the implication by proving the right equation using only the premise. □

5 A Three-Dimensional Model of Spatio-Temporal Predicates

This section presents a different access to an understanding of spatio-temporal predicates. So far, we have considered spatio-temporal objects as functions from time into space, and we have modeled spatio-temporal predicates mainly as sequences of elementary spatio-temporal predicates and spatial predicates. Another very instructive view now is to conceptualize and visualize the temporal evolution of points and regions over time as *purely geometric, three-dimensional objects*, that is, the time axis is regarded as a third geometric dimension. In general, a moving point is then represented as a set of curves in the three-dimensional space and an evolving region as a set of volumes [22]. Spatio-temporal relationships then correspond to relationships between three-dimensional objects which can be described by three-dimensional topological predicates.

In Section 3.2, we have mentioned that Egenhofer's two-dimensional topological predicates can be generalized to the third dimension. This enables us to compare our spatio-temporal predicates on evolving regions with the three-dimensional topological predicates on volumes. As mentioned before, Egenhofer's predicates operate on connected and regular closed two-dimensional regions without holes. These type restrictions exert influence on the definition of volumes as the arguments of three-dimensional topological predicates: Volumes also have to be connected and regular closed and are not permitted to have holes. The consequence is that we can consider only those three-dimensional topological predicates that are associated with this subset of volumes. Accordingly, we cannot deal with general evolving regions but only with a subset of them.

5.1 Correspondence between Spatio-Temporal and 3D Objects

In the following, we focus on the correspondences between volumes and evolving regions and between their three-dimensional and spatio-temporal relationships. From Section 3.3, we know that an evolving region, which is a temporal function, is upper semicontinuous and only has a finite number of temporal change points (discontinuities). The number of temporal change points determines the number of components of an evolving region and shall be given by the function ν. We are only interested in evolving regions with a single component since then we can compare their spatio-temporal relationships with three-dimensional topological relationships that can only be applied to one-component volumes.

To express this formally, we first need some concepts from topology [23]. Let X be a set, and let $\mathcal{T} \subseteq \mathcal{P}(X)$ be a subset of the power set of X. The pair $(X, \mathcal{T})$ is called a *topological space*, if the following three axioms are satisfied: 1) $X \in \mathcal{T}, \emptyset \in \mathcal{T}$, 2) $U, V \in \mathcal{T} \Rightarrow U \cap V \in \mathcal{T}$, and 3) $\mathcal{S} \subseteq \mathcal{T} \Rightarrow \bigcup_{A \in \mathcal{S}} A \in \mathcal{T}$. $\mathcal{T}$ is called a *topology* for X. The elements of $\mathcal{T}$ are called *open sets*, their complements in X *closed sets*. Frequently, when no confusion can arise, $\mathcal{T}$ is omitted, and X denotes a topological space.

Let X, Y be topological spaces. A function $f : X \to Y$ is *continuous* if, for each open subset U of Y, the inverse image $f^{-1}(U) = \{x \in X \mid f(x) \in U\}$ is open in X. A function $f : X \to Y$ is called a *homeomorphism* if it is bijective and continuous, and $f^{-1} : Y \to X$ is continuous. If such a function exists, X and Y are said to be *homeomorphic* or *topologically equivalent* spaces.

Let (X, d) be a metric space [23] with a distance function d as the metric on X. Then, $D(x, \epsilon) = \{y \mid y \in X, d(x, y) \leq \epsilon\}$ is called the *closed ball* with center $x \in X$ and radius $\epsilon > 0$. In our case, since $\mathbb{R}^3$ is even a Euclidean space (as a special instance of a metric space), we can take the Eulidean metric for d described by the Pythagorean distance function

$$d(x, y) = \sqrt{\sum_{i=1}^{3} (x_i - y_i)^2}$$

with $x = (x_1, x_2, x_3), y = (y_1, y_2, y_3) \in \mathbb{R}^3$.

The following definition is based on these concepts. Let $V \subseteq \mathbb{R}^3$, and let $\max_z(V) = \max\{z \mid (x, y, z) \in V\}$ and $\min_z(V) = \min\{z \mid (x, y, z) \in V\}$. We appropriately limit the set of permitted volumes to the type

$$\begin{aligned} volume^c = \{V \subseteq \mathbb{R}^3 \mid\ & V \text{ is topologically equivalent} \\ & \text{to a closed ball in } \mathbb{R}^3, \\ & \forall\ \min_z(V) \leq z \leq \max_z(V) \\ & : \{(x, y) \mid (x, y, z) \in V\} \in region\} \end{aligned}$$

which describes those volumes that are connected, regular closed, and do not have holes, and where each cut through the volume and parallel to the xy-Euclidean plane yields a region. This definition expresses the functional relationship between a z-coordinate of V and the pertaining collection of (x, y)-coordinates that compose a region.

We are now able to formulate the abstraction function α between a volume and an evolving region as a "geometry-to-history" transformation

$$\alpha : volume^c \to \tau(region).$$

Let $V \in volume^c$. Then, $\alpha(V)$ can be defined as

$$\begin{aligned} \alpha(V) = \{(z, r) \mid\ & z \in time, \min_z(V) \leq z < \max_z(V), \\ & r = \{(x, y) \mid (x, y, z) \in V\}\}. \end{aligned}$$

Since each z is mapped to exactly one region, this defines a function. Note that the region at $\max_z(V)$ does not belong to $\alpha(V)$ since $dom(\alpha(V))$ has to be a right half-open time interval according to the definition of a spatio-temporal object (see Section 3.3). Due to the construction process, the whole mapping is upper semicontinuous. This means that there is not an entire correspondence between volumes and evolving regions.

Unfortunately, the definition of an element of $volume^c$ is still too general since it allows volumes to have plateaus parallel to the Euclidean plane that do not coincide with the top surface and that are interpreted as discontinuities in the corresponding evolving regions. As an example, consider two cylinders where the bottom surface of the first cylinder stands partially or completely on the top surface of the second cylinder and where both surfaces are not identical. This configuration models a volume with a single component topologically equivalent to the closed ball, but the corresponding evolving region has two instead of one component. Hence, we define the type

$$volume' = \{V \in volume^c \mid \nu(\alpha(V)) = 1\}$$

which finally describes those volumes we are dealing with.

5.2 Basic Spatio-Temporal versus 3D Topological Predicates

A comparison of three-dimensional topological predicates on volumes and basic spatio-temporal predicates on evolving regions reveals two main reasons for their partially lacking correspondence, as we will see below. The first reason is the different structure at the top of a volume and an evolving region, respectively. The second reason is that some three-dimensional predicates are existentially quantified, whereas their spatio-temporal counterparts are universally quantified and, hence, much stricter.

As mentioned before, for the topological relationships between volumes, we employ the 9-intersection model (Section 3.2) which is here applied to the third dimension and which yields the same collection of eight topological predicates as in the two-dimensional case for regions. We use the notation that, for a two-dimensional spatial predicate p (for example, $inside$), the term p^{3D} (for example, $inside^{3D}$) denotes the corresponding three-dimensional spatial predicate and that the term P (for example, $Inside$) denotes the corresponding basic spatio-temporal predicate. Let $V_1, V_2 \in volume'$. The following lemmas show the relationships between three-dimensional and spatio-temporal predicates:

Lemma 12. $Equal(\alpha(V_1), \alpha(V_2)) \Leftrightarrow equal^{3D}(V_1, V_2)$.

Proof. If $Equal(\alpha(V_1), \alpha(V_2))$ holds, then according to its default aggregation behavior for all times

$$t \in T = dom(\alpha(V_1)) \cup\ dom(\alpha(V_2))$$

(that is, $dom(\alpha(V_1)) = dom(\alpha(V_2))$), the corresponding spatial predicate $equal$ holds for $\alpha(V_1)(t)$ and $\alpha(V_2)(t)$ which are both regions. The time interval T is mapped to a distinguished z-interval Z for the two corresponding volumes V_1 and V_2, which are extended by a top surface. For each coordinate of Z, the respective regions of V_1 and V_2 have the relation $equal$. We know that outside of T, respectively Z, both evolving regions, respectively volumes, are undefined so that no two-dimensional spatial predicate can be valid at such a time t or for such a coordinate z. A comparison with the predicate-specific intersection matrices (Section 3.2, Table 1) shows that corresponding boundaries, interiors, and exteriors of V_1 and V_2 intersect and that all the other intersections yield false. Consequently, $equal^{3D}$ holds.

Conversely, if V_1 and V_2 satisfy the relation $equal^{3D}$, their point sets are equal, and they, in particular, share the same z-interval Z. The relation $equal$ holds for any two regions which we obtain as cuts parallel to the Euclidean plane at all coordinates of Z. The fact that the top surfaces of V_1 and V_2 have no correspondences in $\alpha(V_1)$ and $\alpha(V_2)$ causes no problems because they are omitted in both evolving regions and not taken into account there. Interval Z is mapped to a time interval T, and for all times $t \in T$ predicate $equal$ holds for $\alpha(V_1)(t)$ and $\alpha(V_2)(t)$. For all coordinates, $z \notin Z$ both V_1 and V_2 are undefined so that no two-dimensional spatial predicate can hold there. Consequently, for all times $t \notin T$ both $\alpha(V_1)$ and $\alpha(V_2)$ are undefined and no two-dimensional spatial predicate holds at these instants. Hence, $Equal$ holds for $\alpha(V_1)$ and $\alpha(V_2)$. □

Lemma 13.

$$P(\alpha(V_1), \alpha(V_2)) \Rightarrow p^{3D}(V_1, V_2)$$
$$\forall\ p \in \{meet, overlap, coveredBy, covers\}.$$

Proof. If $P(\alpha(V_1), \alpha(V_2))$ holds, then according to the default aggregation behavior of P for all times $t \in T = \gamma(dom(\alpha(V_1)), dom(\alpha(V_2))$ with $\gamma = \cup$ for $P = Meet$ and $P = Overlap$, $\gamma = \pi_1$ for $P = CoveredBy$, and $\gamma = \pi_2$ for $P = Covers$, the corresponding spatial predicate p holds for $\alpha(V_1)(t)$ and $\alpha(V_2)(t)$, which are both regions. The time interval T is mapped to a distinguished z-interval Z for the two corresponding volumes V_1 and V_2, which have been extended by a top surface. For each coordinate of Z, the respective regions of V_1 and V_2 have the relation p. For $Meet$ and $Overlap$, we know that outside of T, respectively Z, both evolving regions, respectively volumes, are undefined so that no two-dimensional spatial predicate can be valid at such a time t or for such a coordinate z. For $CoveredBy$ and $Covers$ outside of T, respectively Z, at least one evolving region, respectively volume, is undefined so that no two-dimensional spatial predicate can be valid at such a time t or for such a coordinate z. A comparison with the predicate-specific intersection matrices, which is not described here in detail, shows that the four predicates $p^{3D}(V_1, V_2)$ fulfill their intersection matrices. Consequently, all these predicates p^{3D} hold. □

The inverse implication does not hold. The reason is that such a predicate p^{3D} does not require that relation p is valid for all coordinates $z \in Z$ but only for at least one z-coordinate. That is, the predicates p^{3D} and p under consideration have an existential nature and are not as strict as the predicates P which are universally quantified. For example, the spatio-temporal predicates $Meet$ and $Disjoint \triangleright meet \triangleright Disjoint$ correspond to the same three-dimensional predicate $meet^{3D}$.

Lemma 14.

$$P(\alpha(V_1), \alpha(V_2)) \Leftarrow p^{3D}(V_1, V_2)$$
$$\forall\ p \in \{disjoint, inside, contains\}$$

Proof. For $disjoint^{3D}$ let

$$Z = \{z \in \mathbb{R} \mid \exists\ x, y \in \mathbb{R} : (x, y, z) \in V_1 \cap V_2\}.$$

If $disjoint^{3D}(V_1, V_2)$ holds, then for each $z \in Z$, $disjoint$ holds for those regions obtained as cuts parallel to the Euclidean plane from V_1 and V_2. Interval Z is mapped to a time interval T and, for all times $t \in T$, predicate $disjoint$ holds for $\alpha(V_1)(t)$ and $\alpha(V_2)(t)$. For all coordinates $z \notin Z$, either V_1 or V_2, or both, are undefined so that no two-dimensional spatial predicate can hold there. Consequently, for all times $t \notin T$ either $\alpha(V_1)$ or $\alpha(V_2)$, or both are undefined and no two-dimensional spatial predicate holds at these instants. Hence, $Disjoint$ holds for $\alpha(V_1)$ and $\alpha(V_2)$.

For $inside^{3D}$ let

$$Z = \{z \in \mathbb{R} \mid \exists\ x, y \in \mathbb{R} : (x, y, z) \in V_1\}.$$

If $inside^{3D}(V_1, V_2)$ holds, then for each $z \in Z$, $inside$ holds for those regions obtained as cuts parallel to the Euclidean plane from V_1 and V_2. Interval Z is mapped to a time interval T and, for all times $t \in T$, predicate $inside$ holds for $\alpha(V_1)(t)$ and $\alpha(V_2)(t)$. For all coordinates $z \notin Z$, V_1 is undefined so that no two-dimensional spatial predicate can hold there. Consequently, for all times $t \notin T$, $\alpha(V_1)$ is undefined and no two-dimensional spatial predicate holds at these instants. Hence, $Inside$ holds for $\alpha(V_1)$ and $\alpha(V_2)$. The case for $contains^{3D}$ is proved analogously. □

The inverse implication does not hold. We demonstrate this by examples. For the disjointedness relationship, assume that the boundary of the top surface of a cylinder V_1 touches the boundary of the bottom surface of another cylinder V_2 in a point and that the remaining parts of the surface regions and of the cylinders do not have any other points in common. Then, $disjoint^{3D}(V_1, V_2)$ yields false because V_1 and V_2 meet, but $Disjoint(\alpha(V_1), \alpha(V_2))$ yields true since the top surface of V_1 does not have a correspondence in $\alpha(V_1)$ so that $\alpha(V_1)$ and $\alpha(V_2)$ are disjoint. For the inside relationship, assume that cylinder V_1 is contained in cylinder V_2 and that, in particular, the bottom

surface of V_1 is properly contained in the bottom surface of V_2. Then, $inside^{3D}(V_1, V_2)$ yields false because V_1 is only covered by V_2, that is, $coveredBy^{3D}(V_1, V_2)$ holds. But, since at that instant the corresponding region (bottom surface) of $\alpha(V_1)$ is inside the corresponding region (bottom surface) of $\alpha(V_2)$, $Inside(\alpha(V_1), \alpha(V_2))$ holds.

5.3 Complex Spatio-Temporal Predicates

In summary, we can find a three-dimensional correspondence for each basic spatio-temporal predicate except for $Disjoint$, $Inside$, and $Contains$. Due to the introduction of combinators, we can introduce much more (complex) spatio-temporal predicates, and our predicates allow a much more fine-grained characterization of the evolution of spatio-temporal constellations than the three-dimensional predicates yield on volumes. To demonstrate this, here are some examples of complex spatio-temporal predicates:

$$
\begin{array}{rcl}
Touch & := & Disjoint \triangleright meet \triangleright Disjoint \\
Snap & := & Disjoint \triangleright Meet \\
Release & := & Meet \triangleright Disjoint \\
Bypass & := & Snap \triangleright Release \\
Excurse & := & Meet \triangleright Disjoint \triangleright Meet \\
Into & := & meet \triangleright Overlap \triangleright coveredBy \\
OutOf & := & Into^{\leftarrow} \\
Enter & := & Disjoint \triangleright Into \triangleright Inside \\
Leave & := & Enter^{\leftarrow} \\
Cross & := & Enter \triangleright Leave \\
Melt & := & Disjoint \triangleright meet \triangleright Overlap \triangleright Equal \\
Separate & := & Melt^{\leftarrow} \\
Spring & := & equal \triangleright Overlap \triangleright meet \triangleright Disjoint \\
Graze & := & Disjoint \triangleright meet \triangleright Overlap \triangleright \\
 & & (CoveredBy \triangleright Overlap)^{*} \triangleright meet \triangleright Disjoint.
\end{array}
$$

These definitions should be self-explanatory. (Of course, the chosen names depend on the applications using them.) On the other hand, they allow us to define nuances like the two predicates $Separate$ and $Spring$ which only differ in the first elements of their sequences.

Next, we investigate by which three-dimensional predicate our examples for complex spatio-temporal predicates can be characterized. The following tables show that several distinct spatio-temporal predicates lead to the same three-dimensional predicate. In a certain sense, this reveals some kind of coarseness of Egenhofer's 9-intersection model giving thus only a rough characterization of the nature of topological relationships between two volumes. Even rather distinct complex spatio-temporal predicates like $Enter$ and $Cross$ amount to the same three-dimensional predicate.

ST pred.	3D pred.
$Into$	$overlap^{3D}$
$OutOf$	$overlap^{3D}$
$Enter$	$overlap^{3D}$
$Leave$	$overlap^{3D}$
$Cross$	$overlap^{3D}$
$Melt$	$overlap^{3D}$
$Separate$	$overlap^{3D}$
$Spring$	$overlap^{3D}$
$Graze$	$overlap^{3D}$

ST pred.	3D pred.
$Touch$	$meet^{3D}$
$Bypass$	$meet^{3D}$
$Snap$	$meet^{3D}$
$Release$	$meet^{3D}$
$Excurse$	$meet^{3D}$

ST pred.	3D pred.
$\overline{p}$	p^{3D}
$\overleftarrow{p}$	p^{3D}
$\overrightarrow{p}$	p^{3D}
$\overleftrightarrow{p}$	p^{3D}
$\hat{p}$	p^{3D}
$\dot{p}$	p^{3D}

The third table, where

$$
\begin{aligned}
p \in \{&disjoint, equal, meet, overlap, \\
&coveredBy, covers, inside, contains\},
\end{aligned}
$$

shows the obvious necessity that all variants of elementary spatio-temporal predicates that have been produced by different temporal quantifiers have to be assigned to the same three-dimensional predicate. An exception is the aforementioned predicate $Disjoint := \overline{disjoint}$ which cannot be characterized by the three-dimensional predicate $disjoint^{3D}$.

One could further consider the comparison of the spatio-temporal predicates between a moving point and an evolving region with the 19 possible three-dimensional topological predicates between a three-dimensional curve and a volume and the comparison of the spatio-temporal predicates between two moving points with the 33 possible three-dimensional topological predicates between two three-dimensional curves. For lack of space, we will not describe these comparisons in detail here due to the large numbers of possible three-dimensional topological predicates but only delineate some observations.

First, we can observe that these topological predicates cannot be expressed by our elementary spatio-temporal predicates. In the moving point/evolving region case, we only have the three elementary spatio-temporal predicates $Disjoint$, $Meet$, and $Inside$; in the moving point/moving point case, we only have the two elementary spatio-temporal predicates $Disjoint$ and $Meet$.

Second, this does not mean that we cannot formulate spatio-temporal predicates corresponding to the 19, respectively 33, topological predicates. Indeed, we can give for each of these topological predicates a *sequence* of elementary spatio-temporal and spatial predicates. As an example, consider the topological predicate between a three-dimensional curve and a volume where the boundary (end points) of the curve meets the surface of the volume in two three-dimensional points while the interior of the curve is disjoint from the volume. In the spatio-temporal domain, this could be described as $meet \triangleright Disjoint \triangleright meet$. All these sequences are constructed only from elementary spatio-temporal predicates and from spatial predicates.

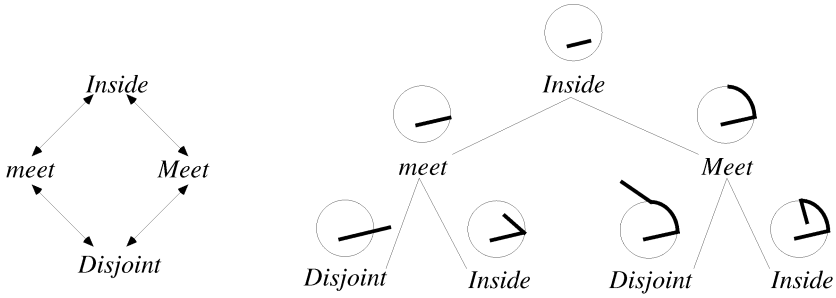

Fig. 3. Development graph for a moving point and an evolving region.

5.4 Query Evaluation

The three-dimensional view of objects and predicates is not only interesting for comparing the expressiveness of three-dimensional topological and spatio-temporal predicates. It can also be exploited for computing the development of two given spatio-temporal objects which is, in particular, relevant for query evaluation [18]. This goal leads to the requirement of efficient computability and has been the main reason not to take a logical approach.

The computation process could work as follows: Consider, for example, the case of an evolving region and a moving point. We take their corresponding three-dimensional geometric counterparts and compute their geometric intersection so that we obtain a sequence of z-intervals as a result together with the information about the corresponding parts of the volume/curve (each of which might be undefined), of their intersection, and of their topological relationship for each z-interval. Either the curve is inside the volume, or the curve runs along the volume border, or the curve is outside the volume. The essential point is that, for each change in the topological relationship, a new z-interval is reported. For topological relationships that are valid only for one single z-coordinate, no (degenerated) interval has to be reported since it and its corresponding predicate can be derived from the adjacent intervals and predicates. The intersection procedures for two volumes and for two curves are analogous. This means that we can compute the evolution of spatial objects by using algorithms from computational geometry and from constructive solid geometry. The validity of a spatio-temporal predicate can then be checked by matching the predicate against the computed evolution.

6 A Canonical Collection of Spatio-Temporal Predicates

In this section, we search for a canonical collection of spatio-temporal predicates from which more complex ones can be composed. Based on the conceptual neighborhood graph described in Section 3.2, we can construct the related, so-called *development graph* which expresses possible topological changes of spatio-temporal objects *over time*. Each vertex is labeled either with a spatial, that is, an instant, predicate (like $meet$), or with an elementary spatio-temporal predicate (like $Disjoint$). Hence, each vertex models a time point or a temporal duration in which this predicate is valid. An edge represents the sequence operator $\triangleright$, that is, (v, w) stands for $v \triangleright w$. A path $(v_1, v_2, \ldots, v_n)$ within the graph describes a possible temporal development $v_1 \triangleright v_2 \triangleright \ldots \triangleright v_n$ of topological relationships between two spatio-temporal objects.

Before we consider the temporal evolution of two evolving regions (the region/region case), we will first deal with an easier case and look at the temporal evolution of a moving point and an evolving region (the point/region case); we will also shortly mention the point/point case. We start with a definition of the development graph.

Definition 10 (Point/Region Development Graph). *The point/region development graph is defined as $DG = (V, E)$ such that*

1. $V = V_{ID} \cup V_{mM}$ *where* $V_{ID} = \{Inside, Disjoint\}$, $V_{mM} = \{meet, Meet\}$.
2. $E = V_{ID} \times V_{mM} \cup V_{mM} \times V_{ID}$.

In summary, we obtain four symmetrical, bidirectional relationships. Considering paths in the development graph, we are not interested in all of them. The reasons are that there exist infinitely many paths due to cycles and that we are searching for a canonical collection of spatio-temporal predicates. Hence, we have to appropriately restrict the set of all possible paths.

A first restriction is to forbid infinite cycles since we are only interested in paths of finite length. Moreover, we tighten this condition according to the following observation. If a path properly contains an instant predicate (like, $meet$) together with its corresponding basic spatio-temporal predicate (like, $Meet$), we can regard this situation as a reoccurrence of a topological relationship. The only difference relates to the temporal duration of the topological relationship. This leads to the definition of a $quasi\text{-}cycle$.

Definition 11 (Quasi-Cycle). *A quasi-cycle of a development graph $DG = (V, E)$ is a path $v = (v_1, v_2, \ldots, v_n)$ in DG such that v is a cycle or such that v_1 is a spatial predicate and v_n is its corresponding spatio-temporal predicate, or vice versa.*

Definition 12 (Development Path). *A development path of a development graph DG is a path in DG which does not properly contain any quasi-cycles.*

In the point/region case, quasi-cycles can only occur between $meet$ and $Meet$. Fig. 3 shows the development graph for the point/region case on the left side and the seven possible development paths starting with $Inside$ on the right side.

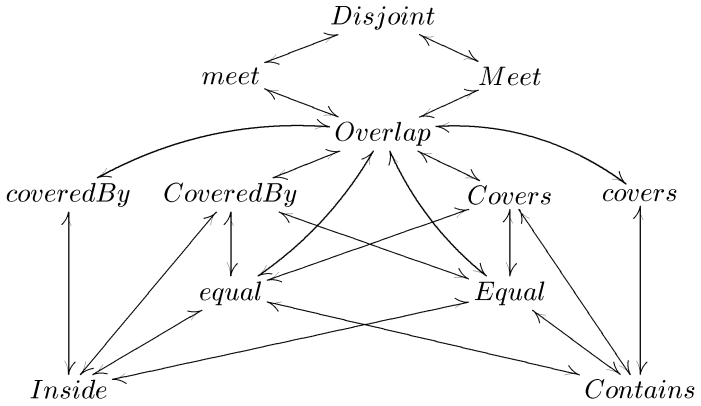

Fig. 4. Development graph for two moving regions.

Since the development graph is symmetric in the point/region case (that is, each of the four vertices can be selected as the start vertex of a path and we obtain the same isomorphic graph), we obtain a total of 28 paths. This means, that there are 28 distinct temporal evolutions of topological relationships between a moving point and an evolving region without repetitions. For each alternative, we could define an own spatio-temporal predicate. In the point/point case, we obtain 13 possible development paths.

Now, we consider the region/region case where the development graph looks, as shown in Fig. 4.

In this case, the number of development paths amounts to 2,198. The large numbers of possible predicates (13 in the point/point case, 28 in the point/region case, and 2,198 in the region/region case) impedes an arrangement of a canonical collection of spatio-temporal predicates. Hence, either the user is furnished with an appropriate, small, application-specific set of predicates, or the user is allowed to use spatial and spatio-temporal predicates as a language and to construct new predicates according to his needs.

To support the user, we have contributed to a solution of these problems (see also Section 2.2) by proposing an extension of the well-known query language SQL to a spatio-temporal query language called *STQL* [18] and by providing a *visual query language* [20], [21]. STQL enables us to textually formulate spatio-temporal queries that relate to developments of spatial objects over time and that are hence based on spatio-temporal predicates. Moreover, a macro-definition mechanism allows the user to compose more complex spatio-temporal predicates from more elementary ones. In a convenient and intuitive manner, the proposed visual language allows the graphical specification of a development, which is then, for example, translated into a sequence of spatio-temporal and spatial predicates. The main idea is to represent spatio-temporal objects in a two-dimensional way by their trace. The intersections of these traces with other objects are then interpreted and translated into the corresponding predicate sequences.

7 Conclusions

We have presented a framework for the specification of spatio-temporal predicates which is based on an integrated view on space and time and on an algebraic data model of spatio-temporal objects. We have shown how developments of spatial relationships can be concisely expressed by building sequences of elementary spatio-temporal and spatial predicates. The compositional design of the framework encourages the modular, structured, and step-by-step creation of complex predicates.

Having investigated a canonical set of spatio-temporal predicates, we must conclude that it is not practical to devise one such set because there are too many predicates that can be considered different. Instead, either an application-dependent subset of predicates is given to the user of a prospective database system, or the (advanced) user is provided with combinators to build up his/her own predicates.

Future work will deal with the issue of how to implement and evaluate spatio-temporal predicates.

References

[1] J.F. Allen, "Maintaining Knowledge about Temporal Intervals," *Comm. ACM*, vol. 26, pp. 832-843, 1983.

[2] J.F. Allen, "Towards a General Theory of Action and Time," *Artificial Intelligence*, vol. 23, pp. 123-154, 1984.

[3] A. Del Bimbo, E. Vicario, and D. Zingoni, "Symbolic Description and Visual Querying of Image Sequences Using Spatio-Temporal Logic," *IEEE Trans. Knowledge and Data Eng.*, vol. 7, no. 4, pp. 609-621, Aug. 1995.

[4] R.S. Bird, *Introduction to Functional Programming Using Haskell*. London, UK: Prentice-Hall Int'l, 1998.

[5] M.H. Böhlen, C.S. Jensen, and B. Skjellaug, "Spatio-Temporal Database Support for Legacy Applications," *Proc. ACM Symp. Applied Computing*, pp. 226-234, 1998.

[6] T.S. Cheng and S.K. Gadia, "A Pattern Matching Language for Spatio-Temporal Databases," *Proc. ACM Conf. Information and Knowledge Management*, pp. 288-295, 1994.

[7] E. Clementini, P. di Felice, and P. van Oosterom, "A Small Set of Formal Topological Relationships Suitable for End-User Interaction," *Proc. Third Int'l Symp. Advances in Spatial Databases*, pp. 277-295, 1993.

[8] J. Clifford and A. Croker, "The Historical Relational Data Model (HRDM) Revisited," *Temporal Databases: Theory, Design, and Implementation*, pp. 6-27, 1993.

[9] J. Clifford, A. Croker, and A. Tuzhilin, "On the Completeness of Query Languages for Grouped and Ungrouped Historical Data Models," *Temporal Databases: Theory, Design, and Implementation*, pp. 496-533, 1993.

[10] Z. Cui, A.G. Cohn, and D.A. Randell, "Qualitative and Topological Relationships in Spatial Databases," *Proc. Third Int'l Symp. Advances in Spatial Databases*, pp. 296-315, 1993.

[11] M.J. Egenhofer, "Definitions of Line-Line Relations for Geographic Databases," *Proc. 16th Int'l Conf. Data Eng.*, pp. 40-46, 1993.

[12] M.J. Egenhofer and K.K. Al-Taha, "Reasoning about Gradual Changes of Topological Relationships," *Proc. Int'l Conf. GIS—From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning in Geographic Space,* pp. 196-219, 1992.
[13] M.J. Egenhofer and R.D. Franzosa, "Point-Set Topological Spatial Relations," *Int'l J. Geographical Information Systems,* vol. 5, no. 2, pp. 161-174, 1991.
[14] M.J. Egenhofer and D. Mark, "Modeling Conceptual Neighborhoods of Topological Line-Region Relations," *Int'l J. Geographical Information Systems,* vol. 9, no. 5, pp. 555-565, 1995.
[15] M. Erwig, R.H. Güting, M. Schneider, and M. Vazirgiannis, "Abstract and Discrete Modeling of Spatio-Temporal Data Types," *Proc. Sixth ACM Int'l Symp. Advances in Geographic Information Systems,* pp. 131-136, 1998.
[16] M. Erwig, R.H. Güting, M. Schneider, and M. Vazirgiannis, "Spatio-Temporal Data Types: An Approach to Modeling and Querying Moving Objects in Databases," *GeoInformatica,* vol. 3, no. 3, pp. 269-296, 1999.
[17] M. Erwig and B. Meyer, "Heterogeneous Visual Languages—Integrating Visual and Textual Programming," *Proc. 11th IEEE Symp. Visual Languages,* pp. 318-325, 1995.
[18] M. Erwig and M. Schneider, "Developments in Spatio-Temporal Query Languages," *Proc. IEEE Int'l Workshop Spatio-Temporal Data Models and Languages,* pp. 441-449, 1999.
[19] M. Erwig and M. Schneider, "The Honeycomb Model of Spatio-Temporal Partitions," *Proc. Int'l Workshop Spatio-Temporal Database Management,* pp. 39-59, 1999.
[20] M. Erwig and M. Schneider, "Visual Specifications of Spatio-Temporal Developments," *Proc. 15th IEEE Symp. Visual Languages,* pp. 187-188, 1999.
[21] M. Erwig and M. Schneider, "Query-By-Trace: Visual Predicate Specification in Spatio-Temporal Databases," *Proc. Fifth Int'l Federation for Information Processing Conf. Visual Databases,* 2000.
[22] M. Erwig, M. Schneider, and R. H. Güting, "Temporal Objects for Spatio-Temporal Data Models and a Comparison of Their Representations," *Proc. Int'l Workshop Advances in Database Technologies,* pp. 454-465, 1998.
[23] S. Gaal, *Point Set Topology.* Academic Press, 1964.
[24] S.K. Gadia and S.S. Nair, "Temporal Databases: A Prelude to Parametric Data," *Temporal Databases: Theory, Design, and Implementation,* pp. 28-66, 1993.
[25] A. Galton, "Towards a Qualitative Theory of Movement," *Proc. Second Int'l Conf. Spatial Information Theory,* pp. 377-396, 1995.
[26] A. Galton, "Continuous Change in Spatial Regions," *Proc. Third Int'l Conf. Spatial Information Theory,* pp. 1-13, 1997.
[27] S. Grumbach, P. Rigaux, and L. Segoufin, "Spatio-Temporal Data Handling with Constraints," *Proc. Sixth ACM Int'l Symp. Advances in Geographic Information Systems,* pp. 106-111, 1998.
[28] R.H. Güting, "Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems," *Proc. Int'l Conf. Extending Database Technology,* pp. 506-527, 1988.
[29] R.H. Güting, M.H. Böhlen, M. Erwig, C.S. Jenssen, N.A. Lorentzos, M. Schneider, and M. Vazirgiannis, "A Foundation for Representing and Querying Moving Objects," *ACM Trans. Database Systems,* vol. 25, no. 1, pp. 1-42, 2000.
[30] R.H. Güting and M. Schneider, "Realm-Based Spatial Data Types: The ROSE Algebra," *VLDB J.* vol. 4, no. 2, pp. 100-143, 1995.
[31] J. Renz and B. Nebel, "On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus," *Artificial Intelligence,* vol. 108, nos. 1-2, pp. 69-123, 1999.
[32] M. Schneider, *Spatial Data Types for Database Systems—Finite Resolution Geometry for Geographic Information Systems.* Springer-Verlag, 1997.
[33] M. Scholl and A. Voisard, "Thematic Map Modeling," *Proc. First Int'l Symp. Large Spatial Databases,* pp. 167-190, 1989.
[34] A. Segev and A. Shoshani, "A Temporal Data Model Based on Time Sequences" *Temporal Databases: Theory, Design, and Implementation,* pp. 248-270, 1993.
[35] M. Stonebraker, "Inclusion of New Types in Relational Database Systems," *Proc. Int'l Conf. Data Eng.,* pp. 262-269, 1986.
[36] M. Stonebraker, B. Rubenstein, and A. Guttman, "Application of Abstract Data Types and Abstract Indices to CAD Data Bases," *Proc. ACM/IEEE Conf. Eng. Design Applications,* pp. 107-113, 1983.
[37] P. Svensson and Z. Huang, "Geo-SAL: A Query Language for Spatial Data Analysis," *Proc. Second Int'l Symp. Large Spatial Databases,* pp. 119-140, 1991.
[38] A.U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, and R. Snodgrass, eds., *Temporal Databases: Theory, Design, and Implementation.* The Benjamin/Cummings Publishing Company, 1993.
[39] Y. Theodoridis, T. Sellis, A. Papadopoulos, and Y. Manolopoulos, "Specifications for Efficient Indexing in Spatiotemporal Databases," *Proc. 10th Int'l Conf. Scientific and Statistical Database Management,* pp. 123-132, 1998.
[40] R.B. Tilove, "Set Membership Classification: A Unified Approach to Geometric Intersection Problems," *IEEE Trans. Computers,* vol. 29, pp. 874-883, 1980.
[41] M.F. Worboys, "A Unified Model for Spatial and Temporal Information," *The Computer J.,* vol. 37, no. 1, pp. 25-34, 1994.
[42] T.S. Yeh and B. de Cambray, "Time as a Geometric Dimension for Modeling the Evolution of Entities: A 3D Approach," *Proc. Int'l Conf. Integrating GIS and Environmental Modeling,* 1993.
[43] T.S. Yeh and B. de Cambray, "Modeling Highly Variable Spatio-Temporal Data," *Proc. Sixth AustraliAsian Database Conf.,* pp. 221-230, 1995.

Martin Erwig received the Diploma degree in computer science from the University of Dortmund, Germany, in 1989. After that, he joined the University of Hagen, Germany, where he received his PhD degree (Dr. rer. nat.) in 1994 and his Habilitation in 1999. Since October 2000, he has been an associate professor at Oregon State University. His research interests are in functional programming, visual languages, and spatial databases. Currently, his work in functional programming is focused on abstract data types, and his research in spatial databases is guided by a functional modeling approach to spatial and spatio-temporal data types.

Markus Schneider received the Diploma degree in computer science from the University of Dortmund, Germany, in 1990, and the PhD degree (Dr. rer. nat.) in computer science from the University of Hagen, Germany, in 1995. He is currently a research assistant (lecturer) at that university. His research interests were first related to the design and implementation of graphical-user interfaces for spatial database systems. Since then, he has worked on the design and implementation of spatial data types (geo-relational algebra, ROSE algebra, realms). Currently, he is interested in research on data modeling for vague or fuzzy spatial objects, on spatio-temporal data modeling, and on partitions in spatial database systems. Moreover, he deals with the design and implementation of data structures and geometric algorithms for these topics.