School voor Informatietechnologie
Kennistechnologie, Informatica, Wiskunde, ICT

Modelling and Querying Spatio-temporal Data

Proefschrift voorgelegd tot het behalen van de graad van
Doctor in de Wetenschappen, richting Informatica
te verdedigen door

Sofie HAESEVOETS

Promotor: Prof. Dr. B. Kuijpers

Juni 2005

Acknowledgements

During the past six years, many people have, in one way or another, contributed to the realization of this thesis. I would like to express my gratitude to them.

First and foremost, to Bart Kuijpers, my advisor, for guiding me through the various stages of the research process, and for sharing with me his knowledge and experience.
To Peter Revesz for inviting me to Nebraska and revealing to me the future of constraint databases. I am also grateful for his and Jan Chomicki's positive response to our follow-up research on the parametric data model.
To Floris Geerts for his encouragement, and enthusiasm for semi-algebraic sets. Chapter 4 describes our joint work.
To the members of our research group at the LUC, for the nice atmosphere and stimulating environment.
To the WNI secretaries for their logistic support and daily "hello".
To the Fund for Scientific Research Flanders for funding my visit to Nebraska.

To my parents, for giving me the opportunity to study and for always supporting me.
To Valérie, for her little notes of encouragement.
To my friends, for helping me to achieve the right balance between labor and leisure.

To Jan, for always encouraging me to broaden my horizon.

Table of Contents

1

Introduction

1.1 Introduction to this Thesis

Most real world phenomena and events have a (possibly time-dependent) spatial extent, *and* are relevant during some period of time, as is illustrated by the following examples. Summer in the northern hemisphere lasts from June until September. A big rain cloud is located above Belgium on May 1st, and above Germany on May 11th. Mr. Meylemans was living in Aarschot from 2001 until 2004. Nevertheless, in database research, spatial databases and temporal databases had been studied already for several years, before the need for spatio-temporal databases was recognized.

In 1993, the "Specialist Meeting on Time in Geographic Space" [23] was organized with the aim of composing a research agenda for the newly-founded spatio-temporal databases field. During that meeting, several remarks were made and questions formulated on the nature of space and time. We concentrate on the research agenda for spatio-temporal data modelling (on an abstract level) and query language design.

Spatio-temporal data modelling questions.

Question 1: Which spatial, temporal and spatio-temporal reference frames should be used? For representing spatial information, sometimes topological relations between spatial objects give sufficient information, while at other occasions metric information is necessary. Is there a difference in using an absolute versus a relative time frame?

Question 2: Can the same rules be applied for the space and time dimensions?
Remark on Question 2: In natural language, time seems to be more restricted than space, and, there exists more terms to describe spatial relationships than to describe temporal relationships.

Spatio-temporal query language design questions.

Question 3: What are relevant spatio-temporal relations and operations?

Question 4: Does there exists a finite set of such relations and operations?

Question 5: Is it sufficient to combine purely spatial operations with purely temporal ones to construct a spatio-temporal query language?

At the moment of the specialist meeting, Question 1 and 3 were not solved yet for spatial databases. In 1994, Paredaens, Van den Bussche and Van Gucht [58] suggested a solution to this problem. They introduced the concept of *genericity* of spatial (constraint) database queries, a concept well-known in classical database theory [11]. A generic query asks only for properties that are shared by "isomorphic" encodings of the same data, or, in other words, the result of a generic query depends only to a certain limited extent on the actual internal representation of the database it is applied to. Whereas Chandra and Harel [11] considered the group of the isomorphisms (that possibly fix some elements of the domain) in the case of relational databases, Paredaens, Van den Bussche and Van Gucht identified different geometrical and topological transformation groups (affinities, isometries, homeomorphisms, etc.) for spatial database applications. Gyssens, Van den Bussche and Van Gucht [43] afterwards defined finite sets of spatial predicates yielding (first-order and computationally) complete languages for the various spatial genericity classes or transformation groups.

The above two articles [58, 43], although they were a big step towards solving Questions 1 and 3, did not receive much attention in the spatio-temporal database community. In fact, those questions were not addressed at all, except for the MurMur [60] project, where the problem of multiple representations for the same data was acknowledged. We generalized the notion of genericity to spatio-temporal databases. This is the main topic of Chapter 4.

We observed that the transformations should first and foremost respect the monotone and unidirectional nature of time, *i.e.*, leave the temporal order of events unchanged. Remark that this is a very natural restriction. It follows that the relevant transformation groups are the product of a group of time-(in)dependent spatial transformations and a group of monotone increasing transformations of the time-component of the spatio-temporal data. Next, we focus on the time-independent spatial transformation groups and study which of them leave different spatial and spatio-temporal properties (like collinearity, distance and orientation) unchanged. We specifically focus on transformation groups that preserve physical properties of spatio-temporal data, like velocity and acceleration. This last one is particularly interesting because it preservers all laws of classical mechanics. We also defined several time-dependent transformation groups as possible genericity classes. They become useful whenever a spatio-temporal event is watched by a moving observer.

We study the notion of spatio-temporal genericity relative to two popular query languages in the constraint model: first-order logic over the reals (FO) and an extension of this logic with a while-loop (FO + While). Both languages are known to be effectively computable (given termination in the case of FO + While-programs) and FO + While is known to be a computationally complete language on constraint databases [71]. First, we show that all the genericity classes are undecidable. We

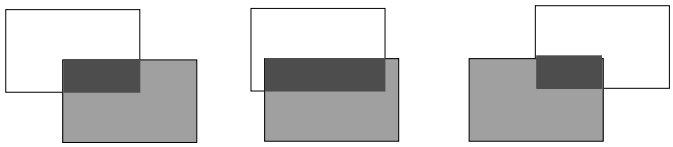


Figure 1.1: Three snapshots are shown of a fixed, light shaded rectangle and a translating white rectangle. Their intersection is the scaling dark shaded rectangle.

show that the considered classes of generic first-order queries are recursively enumerable, however. Hereto, we define first-order point-based languages in which variables are assumed to range over points in $(\mathbb{R}^n \times \mathbb{R})$ and which contain certain point predicates (such as **Between** and **Before**). These point-based languages are shown to be sound and complete for the first-order queries in the considered genericity classes. We have also shown that extensions of these point-based logics with a While-loop give sound and complete programming languages for the computable queries in the different genericity classes.

It turns out, however, that the generic spatio-temporal query languages we proposed in Chapter 4 are not very intuitive, and hence not of great practical use. As the data model we used initially was the constraint model, where spatio-temporal data are semi-algebraic sets, we investigate a simpler data model in Chapter 5. In 1999, Chomicki and Revesz proposed an object-based data model, based on Worboys' *unified objects* [74]. They represented spatio-temporal information as a set of *geometric objects*. A geometric object is a structure containing a spatial reference object, a time interval and a time-dependent transformation function. Chomicki and Revesz also proposed several possible classes of geometric objects.

The first requirement for any spatio-temporal data model is *closure* under certain operations. A class of objects is closed under an operation if the result of applying that operation on any member of that class again is an object of that class. The class of translating rectangles, for example is not closed under intersection. Figure 1.1 shows some snapshots of two translating rectangles and their intersection at several time moments during their movement. The intersection clearly is not a translating, but a "scaling" rectangle. In Chapter 5, we investigate the closure under union, intersection and set difference of a wide range of classes of geometric objects. The conclusion is that only few classes are closed, and hence are suitable to model spatio-temporal data. The most promising is the most general class, that contains all geometric objects that have a triangle as reference object and are transformed by a time-dependent affinity with coefficients that are rational functions of time.

In the geometric model, a spatio-temporal object is a finite union of geometric objects. Here, the need for a normal form becomes clear. For spatial data, triangulations are used as a normal form. Figure 1.2 shows the triangulations of two star-shaped figures. For spatio-temporal data, a formal definition of spatio-temporal partitions was proposed by Erwig and Schneider [29]. However, they do not give a concrete example of such a spatio-temporal partition or partitioning method.

Affine-invariance, or, independence of the choice of coordinate system, is an important concept in computer vision, where recognizing pictures that are the same

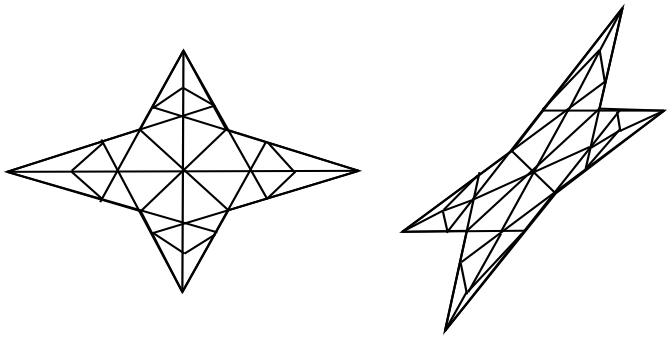


Figure 1.2: The triangulations of a star shape (left) and of an affine transformation of the star shape (right).

up to an affinity is an important issue. In computer graphics, an affine-invariant norm and triangulation have been developed by Nielson [56]. The motivation behind the importance of affine-invariance, is the so-called *weak perspective assumption* [65]. This is the assumption that when an object is repeatedly photographed from different viewpoints, and the object is relatively far away form the camera, that all pictures of the object are affine images of each other. We generalize this assumption for spatio-temporal objects as follows. If a spatio-temporal event is filmed by two moving observers, relatively far away from the event, then both films will be the same up to a time-dependent affinity. For each time moment, another affinity maps the snapshots of the different movies onto each other.

In Chapter 6, we develop a new affine-invariant triangulation method for spatial data and use it to construct an affine-invariant spatio-temporal triangulation of a set of geometric objects. Figure 1.2 shows an affine-invariant spatial triangulation. We show that a finite set of geometric objects of the above mentioned most general class can be converted into a normal form, which is a spatio-temporal partition, with the relaxation that elements of the partition are allowed to share boundaries. This normal form partitions the time domain of a finite set of geometric objects in such a way that for each element in this time partition, the snapshots have the "same" triangulation. As far as we know, this is the first time a concrete spatio-temporal partitioning method is proposed. We analyze the computational and output complexity of both methods.

In Chapter 7, we go back to the constraint model, and consider collections of triangles that can be represented in the constraint model. We develop triangle-based spatial and spatio-temporal affine-generic query languages. Those languages have the same expressive power as the ones we proposed in Chapter 4, but are more intuitive, as they allow the user to express relations between spatial or moving triangles instead of relations on the points that compose some spatio-temporal object. We find out that the language for spatial triangle databases that we develop, containing only one predicate, **PartOf**, can be extended to a language over more general, convex objects. This is in line with the work of Aiello and van Benthem [3, 4] on modal spatial logics. They first propose a topological modal logic over regions, which can express "connectedness" and "parthood". By adding a "convexity" operator (expressed using

a "betweenness" operator), they obtain an affine modal logic. In [4], the authors also motivate the use of finite unions of convex sets as basic elements for spatial reasoning. They argue that it is a very natural way for people to reason about objects. A fork, for example will be described as the union of its prongs and its handle.

1.2 Outline of this Thesis

This thesis is organized as follows. We start, in Section 1.3 with a brief overview on related work. Next, in Chapter 2, we introduce some notation, and give preliminary definitions and properties, that will be used in later chapters. In Chapter 3, the constraint database model is explained and the query languages FO and FO + While are described. We also introduce the concept of genericity. Afterwards, spatio-temporal databases and queries are defined within the constraint model. A hierarchy of genericity classes relevant in the context of spatio-temporal databases is identified, and languages expressing queries generic for those classes are proposed in Chapter 4. After studying spatio-temporal data in its most general form (as permitted within the constraint model) and appropriate query languages, we consider a more concrete data model, motivated by spatio-temporal practice. This object-based *parametric spatio-temporal data model* is introduced in Chapter 5. For a wide range of classes of objects, closure under Boolean set operations is investigated. A normal form for parametric spatio-temporal data is developed next. The first part of Chapter 6 contains a new affine-invariant spatial triangulation method. This triangulation is then extended to a spatio-temporal triangulation, in the second part of the same chapter. Triangle-based logics for both spatial and spatio-temporal databases are proposed in Chapter 7. We end with some concluding remarks in Chapter 8. A summary in Dutch concludes this thesis.

1.3 A Short History of Spatio-temporal Database Research

In this chapter, we describe, in short, the history of the field of spatio-temporal databases. Although we included most of the research important in relation to our work, we make no claim to be exhaustive. A taxonomy of spatio-temporal applications ranging from those that rely on a step-wise constant geometry to applications which need a more complete integration of space and time (like for instance a continuous description of a trajectory) can be found in Erwig et al. [27]. Although the field of spatio-temporal databases and the related field of *temporal GIS (Geographic Information Systems)* have the same origin and the research questions in both fields overlap, they use rather different approaches. In (temporal) GIS on the one hand, the starting point often is a specific problem that cannot be modelled using existing technologies. In spatio-temporal databases, on the other hand, research mostly originates from a more general research question. As a result, the solution is more general than the solutions found in temporal GIS research, but as a consequence it is also more abstract. Within spatio-temporal databases, research already has been done in a broad range of areas, including data modelling, query language design, query processing,

indexing, data mining and dealing with uncertain or incomplete information. As the main topic of this thesis is modelling and querying spatio-temporal databases, we will restrict ourselves to an overview of related research on this topic only.

In the early nineties, when research on spatial databases [1, 9, 26, 38, 41, 64, 66] and GIS was flourishing, questions arose about the role of time in GIS. In 1993, the "Specialist Meeting on Time in Geographic Space" [23] was organized by Research Initiative 10 from the U.S. National Center for Geographic Information and Analysis (NCGIA). This Research Initiative 10 was a research project of the NCGIA titled "Spatio-temporal Reasoning in GIS"; its co-leaders were Egenhofer and Golledge. The main goal of the Specialist Meeting was

"to formulate a research agenda for spatio-temporal reasoning about geographic space. Such reasoning methods should be implementable in a GIS."

A summary of the discussions at the meeting and the identified research questions can be found in a technical report of the NCGIA [23].

One of the participants of the Specialist Meeting was Worboys, who in 1994 proposed the first spatio-temporal database model [74]. He defined a spatio-temporal database as a collection of *unified* objects, that have a spatial extent (a point, line segment or triangle) and a temporal extent (a time interval). As the spatial and temporal extent are completely independent of each other, only piecewise constant movement can be represented. Chomicki and Revesz [14] propose an extension of this model, where objects also have a spatio-temporal component. We will discuss this parametric model in depth in Chapter 5.

In 1996, the CHOROCHRONOS research network [31], a cooperation between ten European institutes was started. The objective of this network was to work together on "Spatial and Temporal Databases". Its main technical goal was

"to study the issues involved in the design and implementation of a STDBMS (Spatio-temporal Database Management System) and to propose an STDBMS architecture".

To achieve this goal, research was subdivided into six tasks:

(i) Ontology and Representation for Space and Time,

(ii) Models and Languages for STDBMS,

(iii) Graphical User Interfaces for Spatio-temporal Information,

(iv) Query Processing in Spatio-temporal Databases,

(v) Storage Structures and Indexing Techniques for Spatio-temporal Databases and

(vi) The Architecture of an STDBMS.

In 2003, an overview of the realizations of CHOROCHRONOS was published [48]. We summarize the main achievements of the CHOROCHRONOS network with respect to the second task, Models and Languages for STDBMS.

Two approaches towards spatio-temporal data modelling were explored, an approach based on data types and one based on constraint databases. We start with the data type approach [39]. A set of base, spatial, temporal and spatio-temporal

data types is proposed. The (two-dimensional) spatial data types are *point*, *points* (a finite set of points), *line* (a finite set of continuous curves) and *region*. Time is to be considered linear and continuous. A type constructor named *moving* exists that, given any type α, yields a mapping from time to α. Examples of types that can be constructed this way are *moving(point)* and *moving(region)*. Next to a set of data types, a set of spatial operations is proposed, that can be *lifted* to spatio-temporal operators. For example, the intersection operator defined on a *region* and a *point* can be lifted in such a way that it can compute the spatio-temporal intersection between a *moving(region)* and a *point*, a *moving(point)* and a *region* or a *moving(region)* and a *moving(point)*. Those operations are embedded in an SQL-like language. The set of data types is fixed, but the user is allowed to define new operations using those data types.

A discrete implementation of this data type approach was proposed. Here, the data type *line* is implemented as a set of line segments, the data type *region* as a collection of polygons with polygonal holes, etc. A *moving(region)* is allowed to change in such a way that its 3-dimensional representation is a polyhedron. As further work the construction of a set of spatio-temporal predicates [30], based on the well-known set of eight topological spatial predicates [25] is proposed. Also, the need for spatio-temporal partitions [28, 29], spatial partitions that are preserved over time, is recognized.

The second approach of the CHOROCHRONOS network towards spatio-temporal data modelling is the constraint database approach. We will explain the constraint database paradigm in Chapter 3. In this approach, the DEDALE data model, spatio-temporal data is represented using linear constraints. It extends the standard language of linear constraints with some additional primitives like *dist* for computing distances and *connect?* for testing connectivity. An SQL-like query language for users is developed on top of the constraint algebra, hiding the data model from the user. The DEDALE model was implemented at INRIA [35]. The developers of DEDALE also introduced the concept of *orthographic dimension* of a constraint relation, which can speed up query evaluation on (spatio-temporal) constraint databases [36, 53].

In 1997, the MOST (Moving Objects Spatio-Temporal) data model was proposed by Sistla, Wolfson, Chamberlain and Dao [68]. In this model, objects can have dynamic attributes, having a *value*, an *update time* and a *function*. This function can be any function f of time for which $f(0) = 0$. The *value* is the value of the function at the current time, and the update time indicates when the value has to be updated. When functions change, for example when an object has a piecewise linear movement, the previous function can be kept by computing the *database state* before the change and store this state in the *database history*. A spatio-temporal query is a predicate over the database history. The FTL (Future Temporal Logic) query language is proposed to query MOST-data. This logic contains two basic future temporal operators *nexttime* and *until*.

In 1999, Chomicki and Revesz [14] proposed the *Parametric Data Model*, an extension of the object-data model of Worboys [74]. Spatio-temporal objects were represented by a spatial reference object and a time domain (analogous to Worboys' spatial and temporal extent [74]) and also a spatio-temporal component, namely, a time-dependent transformation function describing the movement of the spatio-temporal

reference object throughout the time domain. Several classes of such objects were proposed, depending on the type of spatial reference objects and the type of transformation functions. Revesz [10, 63] further developed an algebra for the class of linearly moving rectangles. This algebra includes the specific spatio-temporal operators *buffer*, *compose* and *block*. Both the general model and the query language for linearly moving rectangles are also included in the book "Introduction to Constraint Databases", by Revesz [62]. In Chapter 5, we will give more details on this parametric data model and further investigate it. In Chapter 6, we propose a normal form for the most general class of those parametric objects.

In the same year, the Tripod [34] project emerged, a joint collaboration between researchers in the computer science departments at Keele and Manchester Universities. An existing database system was extended with the spatial types proposed for the ROSE algebra [40], and the temporal types *Instants* and *TimeIntervals*. The Tripod data model can only express discrete changes. Previous states of an object are kept in *histories*.

In 2000, a European project called Multi-representations and Multiple Resolutions in Geographic Databases (MurMur) [60] was started. The participants' goal was

"enhancing GIS (or STDBMS) functionality so that, relying on more flexible representation schemes, users may easily manage information using multiple representations. The added functionality will support multiple coexisting representations of the same real-word phenomena (semantic flexibility), including representations of geographic data at multiple resolutions (cartographic flexibility). This will in particular make possible a semantically meaningful management of multi-scale, integrated, and temporal geo-databases".

MurMur started from an existing spatio-temporal data model, called MADS (Modelling of Application Data with Spatio-temporal features), proposed by Parent, Spaccapietra and Zimányi [61]. This model contains objects, attributes and relationships of several types. Special to MADS is the *perception stamp*, including the viewpoint of the user (public, manager or technician) and the resolution or level of detail of a representation (e.g. 1:2000). These perception stamps allow users to define sub-schemas in a given schema, personalize data types, etc. A two-sorted algebra (the MADS algebra) and two visual query languages were developed to manipulate spatio-temporal data.

At the same time, Chen and Zaniolo [13] proposed SQL^{ST}, a spatio-temporal data model and query language. Here, a moving object is modelled as a series of snapshots, containing *directed triangles*. The query language is based on SQL^{T} [12] for its temporal operators, and a set of spatial operations, like *intersect*, *area*, etc. No real spatio-temporal operations are needed because of the snapshot view.

Also in 2000, Kuijpers, Paredaens and Van Gucht [51] proposed a model and query language for *Movie Databases*. In this proposal, the constraint database approach is used. A movie is modelled as a 2-dimensional semi-algebraic figure that can change in time. A number of computability results concerning movies are given, e.g., it can be decided whether a frame of a movie is only a topologically transformation of another frame, a movie has a finite number of scenes and cuts and these can be effectively computed, etc. Based on these computability results, an SQL-like query language for

movie databases is developed. This query language supports common movie editing operations, like cutting, pasting and selection of scenes.

Another constraint database approach to spatio-temporal databases was proposed in 2001 by Ibarra, Su and Xu [55, 69]. Logical properties of moving objects were considered in connection with queries over such objects using tools from differential geometry. An abstract model was proposed, where object locations can be described as vectors of continuous functions of time. Using this conceptual model, logical relationships between moving objects were examined, and between moving objects and (stationary) spatial objects in the database. These relationships were then characterized in terms of position, velocity, and acceleration. Based on this theoretical foundation, a concrete data model for moving objects was developed, which is an extension of linear constraint databases. The authors also presented a preliminary version of a logical query language for moving object databases.

2

Notations and Preliminaries

In this introductory chapter we, explain notations that will be used throughout this text. Furthermore, we list some preliminary definitions that can be found in literature on databases [2] and mathematics and logic [8, 71, 72], that the Reader might keep in mind when going through the next chapters.

2.1 Notations

The set of the natural numbers will be denoted by $\mathbb{N}$, the set of the real numbers by $\mathbb{R}$ and the n-dimensional real space by $\mathbb{R}^n$. When the $(n+1)$-dimensional real space $\mathbb{R}^{n+1}$ is interpreted as the space in which we consider spatio-temporal objects having an n-dimensional spatial extent, varying over time, we denote it by $(\mathbb{R}^n \times \mathbb{R})$.

Real numbers denoting spatial coordinates are represented by characters a, b, c, a_1, b_1, c_1, a_2, b_2, $c_2, \ldots$, and real numbers denoting moments in time are represented by Greek characters $\tau, \tau_1, \tau_2, \ldots$. Bold characters $\mathbf{a}, \mathbf{a}_1, \mathbf{a}_2, \ldots$ represent n-dimensional spatial vectors. Vectors $(\mathbf{a}, \tau)$ containing mixed spatial and temporal information are denoted $p, q, r, p_1, q_1, r_1, p_2, q_2, r_2, \ldots$. So, we use the notation:

$$p_j = (\mathbf{a}_j, \tau_j) = (a_{j,1}, \ldots, a_{j,n}, \tau_j).$$

Variables that range over real numbers and indicate spatial coordinates will be denoted by characters $x, y, x_1, y_1, z_1, x_2, y_2, \ldots$. For real variables that indicate time coordinates, we use $t, t_1, t_2, \ldots$. Variables that range over vectors in $\mathbb{R}^n$ and that represent spatial information are denoted by bold characters $\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \ldots$, while variables $(\mathbf{x}, t)$ containing such vectors together with a time variable will be represented by characters $u, v, w, u_1, v_1, w_1, u_2, v_2, w_2 \ldots$. To summarize, we use the notation:

$$u_i = (\mathbf{x}_i, t_i) = (x_{i,1}, \ldots, x_{i,n}, t_i).$$

Remark 2.1. Note that multiple lower indices will be all written on the same level, separated by comma's. The coordinates of an n-dimensional spatial vector $\mathbf{x}_i$ are $x_{i,1}$, $x_{1,2}$, ..., $x_{i,n}$, for example.

Let $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ be three points in $\mathbb{R}^n$. The line segment with end points $\mathbf{a}$ and $\mathbf{b}$ will be denoted by $\mathbf{ab}$ and its length by $|\mathbf{ab}|$. The line through the points $\mathbf{a}$ and $\mathbf{b}$, which we call the *carrier of the line segment* $\mathbf{ab}$, will be denoted by $L_{\mathbf{ab}}$. If the three points are not collinear, we denote the triangle that has $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$ as its corner points by $T_{\mathbf{abc}}$.

2.2 Preliminaries

2.2.1 Definitions from Relational Database Theory

Below, we define some basic concepts from database theory [2] that we will use frequently.

Definition 2.2 (**Database**)**.** Assume a countably infinite set $\mathcal{U}$, also called the *underlying domain* and a countably infinite set *rname* of relation names. Each relation name R has an associated *arity*, $ar(R) \in \mathbb{N}$.

A *database schema* is a nonempty, finite set $\sigma = \{R_1, R_2, \ldots, R_k\}$ of relation names. A *relation (instance)* over a relation name R is a (possibly empty) finite set of tuples $d \in \mathcal{U}^{ar(R)}$. A *database (instance)* I over a database schema σ is the set of relation instances over all relation names in σ.

Definition 2.3 (**Query**)**.** A *database query* Q is a partial, computable mapping from an input schema σ_{in} to an output schema σ_{out}. Two queries Q_1 and Q_2 over the input schema σ_{in} are equivalent, denoted $Q_1 \equiv Q_2$, if they have the same output schema and $Q_1(I) = Q_2(I)$ for each database instance I over σ_{in}.

One of the pillars of database theory, is *data independence*, meaning that a database gives an abstract interface to its contents, hiding the internal representation of the data. In other words, the database is essentially unchanged if a permutation is applied to $\mathcal{U}$. An important related property of database queries is *genericity*.

Definition 2.4 (**Genericity**)**.** Let σ_{in} and σ_{out} be database schemas. A mapping Q from σ_{in} to σ_{out} is *generic* if and only if for each database instance I over σ_{in} and each permutation ρ of $\mathcal{U}$, $\rho(Q(I)) = Q(\rho(I))$.

Note that it is allowed that Q explicitly names a finite set C of constants, which are not changed by ρ. In that case, Q is called *C-generic*.

2.2.2 Definitions from Mathematics and Logic

In this subsection, we give some definitions [71] that we will use in Chapter 3, when explaining the *constraint database model*.

Definition 2.5 (**Signature**)**.** A *signature* Ω consists of a set $\mathcal{F}$ of function symbols, a set $\mathcal{P}$ of predicate symbols and a set C of constant symbols. All elements f of $\mathcal{F}$ and all elements P of $\mathcal{P}$ have an associated arity $ar(f) \in \mathbb{N}$ and $ar(P) \in \mathbb{N}$, respectively.

Definition 2.6 (**Structure**). Given a set $\mathcal{U}$ and a signature Ω, an Ω-*structure* $\mathcal{M}$ *over* $\mathcal{U}$, denoted $\langle \mathcal{U}, \Omega \rangle$, is defined by assigning to each $f \in \mathcal{F}$ of arity m, a function $f^{\mathcal{M}} : \mathcal{U}^m \mapsto \mathcal{U}$, to each $P \in \mathcal{P}$ of arity ℓ, an ℓ-ary predicate on $\mathcal{U}$, that is, a set $P^{\mathcal{M}} \subseteq \mathcal{U}^\ell$, and to each $\kappa \in \mathcal{C}$ an element $\kappa^{\mathcal{M}} \in \mathcal{U}$.

We now define, by induction on the structure of its formulas, *first-order logic over a signature* Ω.

Definition 2.7 (**First-order logic over** Ω). Assume a countably infinite set of variables $V = \{\nu_1, \nu_2, \ldots\}$.

(i) A *term* is either a variable from V, a constant from $\mathcal{C}$, or $f(\theta_1, \theta_2, \ldots, \theta_m)$, where $f \in \mathcal{F}$ has arity m and $\theta_1, \theta_2, \ldots, \theta_m$ are terms.

(ii) An *atomic formula* is a formula of the form $\theta = \theta'$ or $P(\theta_1, \theta_2, \ldots, \theta_\ell)$, where $P \in \mathcal{P}$ has arity ℓ and $\theta, \theta', \theta_1, \theta_2, \ldots, \theta_\ell$ are terms.

(iii) A formula in first-order logic over Ω is an atomic formula or a formula of the form $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee \psi$, $\exists\nu\, \varphi$ or $\forall\nu\, \varphi$, where φ and ψ are formulas in first-order logic over Ω and $\nu \in V$.

Given a structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$, a term $\theta(\nu_1, \nu_2, \ldots, \nu_m)$ and elements $d_1, d_2, \ldots, d_m$ of $\mathcal{U}$. The *interpretation of* θ *given* $d_1, d_2, \ldots, d_m$ *in* $\mathcal{M}$, denoted by $\theta^{\mathcal{M}}[d_1, d_2, \ldots, d_m]$, is defined as follows.

(i) If θ is a variable ν_i then $\theta^{\mathcal{M}}[d_1, d_2, \ldots, d_m]$ is d_i $(1 \leq i \leq m)$,

(ii) if θ is a constant symbol κ then $\theta^{\mathcal{M}}[d_1, d_2, \ldots, d_m]$ is $\kappa^{\mathcal{M}}$, and

(iii) if θ is of the form $f(\theta_1, \theta_2, \ldots, \theta_\ell)$ then $\theta^{\mathcal{M}}[d_1, d_2, \ldots, d_m]$ is

$$f^{\mathcal{M}}(\theta_1^{\mathcal{M}}[d_1, d_2, \ldots, d_m], \theta_2^{\mathcal{M}}[d_1, d_2, \ldots, d_m], \ldots, \theta_\ell^{\mathcal{M}}[d_1, d_2, \ldots, d_m]).$$

Given a structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$, a first-order-formula φ with free variables $\nu_1, \nu_2, \ldots, \nu_m$ and the elements $d_1, d_2, \ldots, d_m$ from $\mathcal{U}$, the *satisfaction of* φ *by* $(d_1, d_2, \ldots, d_m)$ *in* $\mathcal{M}$, denoted by $\mathcal{M} \models \varphi[d_1, d_2, \ldots, d_m]$ is defined as follows.

(i) $\mathcal{M} \models (\theta = \theta')[d_1, d_2, \ldots, d_m]$ if $\theta^{\mathcal{M}}[d_1, d_2, \ldots, d_m]$ equals $\theta'^{\mathcal{M}}[d_1, d_2, \ldots, d_m]$,

(ii) $\mathcal{M} \models P(\theta_1, \theta_2, \ldots, \theta_\ell)[d_1, d_2, \ldots, d_m]$ if

$$(\theta_1^{\mathcal{M}}[d_1, d_2, \ldots, d_m], \theta_2^{\mathcal{M}}[d_1, d_2, \ldots, d_m], \ldots, \theta_\ell^{\mathcal{M}}[d_1, d_2, \ldots, d_m]) \in P^{\mathcal{M}},$$

(iii) $\mathcal{M} \models (\neg\varphi)[d_1, d_2, \ldots, d_m]$ if $\mathcal{M} \models \varphi[d_1, d_2, \ldots, d_m]$ does not hold,

(iv) $\mathcal{M} \models (\varphi \wedge \psi)[d_1, d_2, \ldots, d_m]$ if both $\mathcal{M} \models \varphi[d_1, d_2, \ldots, d_m]$ and $\mathcal{M} \models \psi[d_1, d_2, \ldots, d_m]$,

(v) $\mathcal{M} \models (\varphi \vee \psi)[d_1, d_2, \ldots, d_m]$ if either $\mathcal{M} \models \varphi[d_1, d_2, \ldots, d_m]$ or $\mathcal{M} \models \psi[d_1, d_2, \ldots, d_m]$,

Now let ψ be a first-order-formula with free variables $\nu_1, \nu_2, \ldots, \nu_m, \nu_{m+1}$.

(vi) $\mathcal{M} \models (\forall \nu_{m+1}\, \psi)[d_1, d_2, \ldots, d_m]$ if for every element κ of $\mathcal{U}$, we have $\mathcal{M} \models \psi[d_1, d_2, \ldots, d_m, \kappa]$, and

(vii) $\mathcal{M} \models (\exists \nu_{m+1}\, \psi)[d_1, d_2, \ldots, d_m]$ if for some element κ of $\mathcal{U}$, we have $\mathcal{M} \models \psi[d_1, d_2, \ldots, d_m, \kappa]$.

We will indicate that a first-order-formula φ has free variables $\nu_1, \nu_2, \ldots, \nu_\ell$ by writing $\varphi(\nu_1, \nu_2, \ldots, \nu_\ell)$.

Remark 2.8. The set of all first-order-formulas over Ω is denoted FO(Ω). If $\mathcal{M}$ is an Ω-structure, we sometimes also write FO($\mathcal{M}$) for FO(Ω).

Example 2.9. Given the signature (+, ×, <, 0, 1) and the set $\mathbb{R}$ of real numbers, first-order logic over the field of the real numbers will be denoted FO($\mathbb{R}$, +, ×, <, 0, 1). □

Definition 2.10 (**Constraint**)**.** Let Ω be a vocabulary. A *constraint over* Ω is an atomic formula in first-order logic over Ω.

Given a structure $\mathcal{M} = \langle \mathcal{U}, \Omega \rangle$, a set $X \subseteq \mathcal{U}^k$ is called *definable on* $\mathcal{M}$ *with* Ω*-constraints* if it can be obtained as a finite Boolean combination of sets of the form

$$\{(d_1, d_2, \ldots, d_k) \in \mathcal{U}^k \mid \mathcal{M} \models \varphi[d_1, d_2, \ldots, d_k]\},$$

where φ is a constraint over Ω.

We now define an important class of spatial figures, *semi-algebraic* sets. For more mathematical background on those figures and their properties, we refer to [8].

Definition 2.11 (**Semi-algebraic set**)**.** A *semi-algebraic set in* $\mathbb{R}^n$ is a subset of $\mathbb{R}^n$ that can be described as a Boolean combination of sets of the form

$$\{(x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \mid p(x_1, x_2, \ldots, x_n) > 0\},$$

where p is a polynomial in the variables $x_1, x_2, \ldots, x_n$, with integer coefficients.

We immediately point to an important property of semi-algebraic sets:

Property 2.2.1 (**Semi-algebraic sets definable with polynomial constraints**)**.** The semi-algebraic sets are exactly the sets definable on $(\mathbb{R}, +, \times, <, 0, 1)$ with constraints over (+, ×, <, 0, 1).

Therefor, we will sometimes use the expression *polynomial constraints* for the constraints over (+, ×, <, 0, 1).

It is well-known that the set of semi-algebraic constraints is closed under finite unions, intersections and complements. Also, the cartesian product $A \times B$ of two semi-algebraic sets $A \subset \mathbb{R}^n$ and $B \subset \mathbb{R}^m$ is again a semi-algebraic set in $\mathbb{R}^{(n+m)}$. Furthermore, let $A \subset \mathbb{R}^{(n+1)}$ be a semi-algebraic set and $\pi : \mathbb{R}^{(n+1)} \mapsto \mathbb{R}^n$ be the projection on the first n coordinates, then $\pi(A)$ is a semi-algebraic subset of $\mathbb{R}^n$.

Definition 2.12 (**Semi-algebraic function**)**.** Let A be a semi-algebraic subset of $\mathbb{R}^n$. A function $f : A \mapsto \mathbb{R}^m$ is called *semi-algebraic* if its graph

$$\Gamma(f) = \{(\mathbf{x}, \mathbf{y}) \in A \times \mathbb{R}^m \mid \mathbf{x} \in A \text{ and } \mathbf{y} = f(\mathbf{x})\}$$

is a semi-algebraic subset of $\mathbb{R}^{n+m}$.

Now we define semi-algebraic triviality, and give the Triviality Theorem [8, 15, 72], which we will rely on in Chapter 5. Let $A \subseteq \mathbb{R}^n$ be a semi-algebraic set. A continuous semi-algebraic mapping $p : A \to \mathbb{R}^k$ is said to be *semi-algebraically trivial* over a semi-algebraic subset $C \subset \mathbb{R}^k$ if there is a semi-algebraic set F and a semi-algebraic homeomorphism $h : p^{-1}(C) \to C \times F$, such that the composition of h with the projection $C \times F \to C$ is equal to the restriction of p to $p^{-1}(C)$.

Theorem 2.13 (**Semi-algebraic triviality theorem**)**.** Let $A \subseteq \mathbb{R}^n$ be a semi-algebraic set and $p : A \to \mathbb{R}^k$, a continuous semi-algebraic mapping. There is a finite semi-algebraic partition of $\mathbb{R}^k$ into $C_1, \ldots, C_m$ such that p is semi-algebraically trivial over each C_i.

In particular, if $\mathbf{a}$ and $\mathbf{b}$ are in the same C_i, then $p^{-1}(\mathbf{a})$ and $p^{-1}(\mathbf{b})$ are semi-algebraically homeomorphic.

3

The Constraint Database Model

An important limitation of the classical relational database model is that it can only represent a finite amount of data. More specific, a relational database consists of a finite set of tables, each containing a possibly huge, but finite, amount of tuples. Although this is sufficient for a lot of applications, there exists infinite information that we would like to store in a database. Spatial information, for instance, is inherently infinite. Each figure is, essentially, a possibly infinite set of points in a real space $\mathbb{R}^n$. Figure 3.1 shows an example of an infinite set of points in $\mathbb{R}^3$.

A recent and much acclaimed method for effectively representing infinite geometrical figures is provided by the *constraint database model*, that was introduced in 1990 by Kanellakis, Kuper and Revesz [46, 47]. Intensive research on constraint databases has been carried out since, and a few recent books give an overview of the already well-established field [59, 62]. The main contribution of the constraint database paradigm is that it enables us to represent infinite information in a finite way, by storing the (finite) *formulas* describing the infinite information. The information shown in Figure 3.1, for example, can be represented in a finite way using the formula $\{(x, y, t) \in \mathbb{R}^3 \mid x^2+y^2+t^2 \leq 1 \vee (x^2+y^2+(t-2)^2 = 1 \wedge t \leq 5/2) \vee (x^2+y^2+(t-3)^2 = 1 \wedge t > 5/2)\}$.

In this chapter, we give a short introduction to the constraint database model and show how this model can be used for representing spatio-temporal information. Indeed, each object in $\mathbb{R}^n$ moving through time can be seen as a spatial object in the real space $\mathbb{R}^{(n+1)}$. We start with a formal definition of the constraint database model.

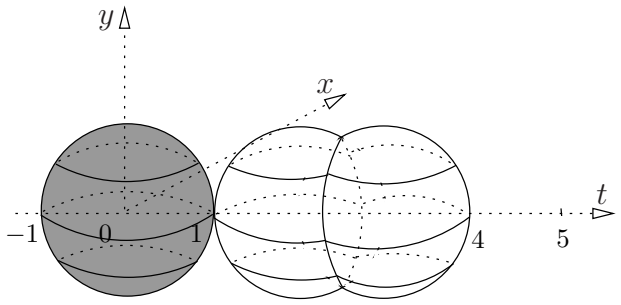


Figure 3.1: An example of a spatial database in $\mathbb{R}^3$.

3.1 The Constraint Database Model

Let $\sigma = \{R_1, R_2, \ldots, R_m\}$ be a finite set of relation names. To each relation name R_i, a natural number is assigned, which we call its *arity* and denote by $ar(R_i)$, $1 \leq i \leq m$.

In the constraint framework, an *intensional relation of arity* k is a quantifier-free formula $\varphi(x_1, x_2, \ldots, x_k)$ in FO(+, ×, <, 0, 1). An *intensional instance over a database schema* σ is a mapping D on σ assigning to each relation name R_i of arity k that appears in σ an intensional relation φ of arity k.

An intensional relation R, given by the formula $\varphi(x_1, x_2, \ldots, x_k)$, represents on the extensional level the subset

$$\{(a_1, a_2, \ldots, a_k) \in \mathbb{R}^k \mid (\mathbb{R}, +, \times, <, 0, 1) \models \varphi[a_1, a_2, \ldots, a_k]\}$$

of $\mathbb{R}^k$. We call this possibly infinite set an extensional relation. In the mathematical literature, these extensional relations are referred to as *semi-algebraic sets* (see Definition 2.11).

An *extensional database instance over a database schema* σ that intentionally is represented by a set of FO(+, ×, <, 0, 1)-formulas $\varphi_{R_1}(x_1, x_2, \ldots, x_{k_1})$, $\varphi_{R_2}(x_1, x_2, \ldots, x_{k_2})$, …, $\varphi_{R_m}(x_1, x_2, \ldots, x_{k_m})$, where $k_i = ar(R_i)$, for $i = 1, \ldots, m$, represents on the extensional level a set of constraint relations (or semi-algebraic sets)

$$\{(a_1, a_2, \ldots, a_{k_i}) \in \mathbb{R}^{k_i} \mid (\mathbb{R}, +, \times, <, 0, 1) \models \varphi_{R_i}[a_1, a_2, \ldots, a_{k_i}]\},$$

for $i = 1, \ldots, m$.

For a database schema σ, we denote the set of its intensional instances by i-inst(σ) and the set of its extensional instances by e-inst(σ).

In the remainder of this text, we will use the terms constraint relation and semi-algebraic set interchangeably, since they refer to the same objects, albeit from different perspectives.

Remark 3.1. The semi-algebraic sets correspond to the subsets of $\mathbb{R}^n$ that can be specified using polynomial constraints (see Property 2.2.1). Therefor, the constraint relations and databases defined above can also be referred to as *polynomial constraint relations* and *polynomial constraint databases*.

Example 3.2. Figure 3.1 gives an example of a constraint relation in $\mathbb{R}^3$. This set can be described as follows: $\{(a,b,c) \in \mathbb{R}^3 \mid a^2+b^2+c^2 \leq 1 \vee (a^2+b^2+(c-2)^2 = 1 \wedge c \leq 5/2) \vee (a^2+b^2+(c-3)^2 = 1 \wedge c > 5/2)\}$. □

Example 3.3. Let $\sigma = \{R_1, R_2\}$, with $ar(R_1) = 2$ and $ar(R) = 1$ be a database schema. Then the instance D, where $R_1^D = \{(a,b) \in \mathbb{R}^2 \mid a^2+b^2 < 1\}$ and $R_2^D = \{a \in \mathbb{R} \mid 0 \leq a \leq 1\}$, is an example of a constraint database over σ that contains the open unit disk and the closed unit interval. □

3.2 Constraint Database Queries

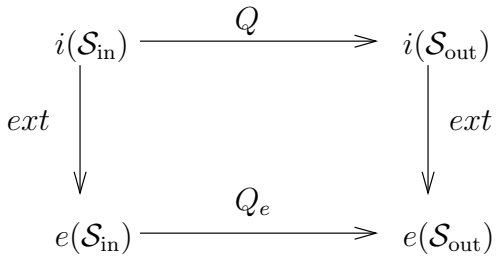
In this section, we define constraint database queries and describe how first-order logic is used as a query language for constraint databases. We also define *generic constraint queries* and give languages for expressing such generic queries.

Note that two intensional relations are called *equivalent* if they define the same extensional relation.

Definition 3.4 (**Constraint database query**)**.** Let σ_{in} and σ_{out} be disjoint constraint database schemas.

An *extensional constraint database query* Q_{ext} of signature $\sigma_{\text{in}} \to \sigma_{\text{out}}$ is a partial, computable mapping from e-inst(σ_{in}) to e-inst(σ_{out}).

An *intensional constraint database query* Q of signature $\sigma_{\text{in}} \to \sigma_{\text{out}}$ is a partial, computable mapping from i-inst(σ_{in}) to i-inst(σ_{out}) that is defined up to equivalence of the formulas appearing in i-inst(σ_{in}) and i-inst(σ_{out}), for which there exists an extensional query Q_e of the same signature such that the diagram



commutes. Here, *ext* is the function that maps an intensional database instance to the corresponding extensional instance.

When we refer to a *constraint database query*, we allude to a query on the intensional level (to which a query on the extensional level corresponds).

Remark 3.5. For the remainder of this chapter, and also for Chapter 4 and Chapter 7, we assume that databases are finitely encoded by systems of polynomial equations and that a specific data structure is fixed (possible data structures are dense or sparse representations of polynomials). The specific choice of data structure is not relevant to the topic of the chapters mentioned above, but we assume that one is fixed. When we talk about computable queries later on, we mean Turing computable with respect to the chosen encoding and data structures.

3.2.1 The Languages FO and FO + While

First-order logic (or FO, in short) over $(\mathbb{R}, +, \times, <, 0, 1)$ has been well-studied as a query language in the context of constraint databases [47, 59].

Definition 3.6 (**First-order logic with polynomial constraints**). Let $\sigma_{\text{in}} = \{R_1, R_2, \ldots, R_m\}$ be a database schema. A *first-order formula over* σ_{in} is a formula $\varphi(x_1, x_2, \ldots, x_k)$ in the first-order language over the vocabulary (+, ×, <, 0, 1, R_1, R_2, …, R_m) (also abbreviated as (+, ×, <, 0, 1, σ_{in})).

Given an extensional instance $D = (D_1, D_2, \ldots, D_m) \in \text{e-inst}(\sigma_{\text{in}})$ and a first-order formula $\varphi(x_1, x_2, \ldots, x_k)$ over the vocabulary (+, ×, <, 0, 1, R_1, R_2, …, R_m). The formula φ, when evaluated on the input D, defines an extensional relation $\varphi(D)$ in the natural way by interpreting variables as ranging over the real numbers. More precisely, if φ has k free variables, then $\varphi(D)$ refers to the set

$$\{(a_1, a_2, \ldots, a_k) \in \mathbb{R}^k \mid (\mathbb{R}, +, \times, <, 0, 1, D_1, D_2, \ldots, D_m) \models \varphi[a_1, a_2, \ldots, a_k]\}.$$

Every extensional query expressible by a FO(+, ×, <, 0, 1)-formula corresponds to an intensional query.

Remark 3.7. In the remainder of this text, we will be less formal when we describe queries and in particular FO(+, ×, <, 0, 1)-queries. We will not always mention the input and output schema explicitly. Furthermore, we will also use a more relaxed formalism and omit reference to intensional and extensional for relations, databases or queries. In most cases, we just refer to formulas when talking about queries. In should be clear that these formulas express queries in the above introduced senses.

Example 3.8. The query returning all points of a 2-dimensional relation R that lie on the x-axis can be expressed in first-order logic by the formula $\varphi(x, y)$

$$R(x, y) \wedge (y = 0).$$

□

The free variables x and y in the above formula are the coordinates of the points in the result of the query. Although their variables range over the real numbers, first-order expressions can be computed effectively. This property is a direct consequence of the fact that there exists a quantifier-elimination procedure for the first-order theory of the closed real fields [70].

The expressive power of first-order logic is limited. Examples of properties of semi-algebraic sets that cannot be expressed in FO(+, ×, <, 0, 1) are parity and connectivity, for instance. Gyssens, Van den Bussche and Van Gucht [43, 71] showed that adding a while-loop to FO(+, ×, <, 0, 1) (this language will be denoted FO + While) drastically increases its expressive power; the language FO + While is computationally complete on polynomial constraint databases. This means that for every partial computable (in the sense of Remark 3.5) query Q on constraint databases, there exists an expression φ in FO + While such that for each constraint database instance D,

$\varphi(D)$ is defined if and only if $Q(D)$ is defined, and in this case $\varphi(D)$ and $Q(D)$ are equal.

We give a more precise definition of FO + While-queries, often called *programs*. The use of similar languages will be illustrated in Section 4.3. We also refer to [43, 71] for illustrations.

Definition 3.9 (**First-order logic extended with a while loop**)**.** Let σ be a constraint database schema. Syntactically, a *program* in the language FO(+, ×, <, 0, 1, σ)+While is a finite sequence of *statements* and *while-loops*. It is assumed there is a sufficient supply of new relation variables, each with an appropriate arity.

(i) Each statement has the form

$$R := \{(x_1, \ldots, x_k) \mid \varphi(x_1, \ldots, x_k)\};$$

Here, R is a relation variable with assigned arity k (the variables x_i range over $\mathbb{R}$) and φ is a formula in FO(+, ×, <, 0, 1, σ'), where σ' is the set of relation names containing the elements of σ together with the relation variables introduced in previous statements of the program.

(ii) A while-loop has the form

while φ **do**
P
end while

where P is a program and φ is a sentence in FO(+, ×, <, 0, 1, σ'), where σ' is again the set of relation names containing the elements of σ together with the relation variables introduced in previous statements of the program.

(iii) One of the relation names occurring in the program is designated as the output relation and is named R_{out}.

Semantically, a program in the query language FO + While expresses a constraint database query as soon as R_{out} is assigned a return value. The execution of an FO(+, ×, <, 0, 1, σ)+While-program applied to an input database is performed step-by-step. A statement is executed by first evaluating the FO(+, ×, <, 0, 1, σ)-formula on the right hand side on the input database together with the new relations resulting from previous statements. Next, the result of the evaluation of the right hand side is assigned to the relation variable on the left-hand side. The effect of a while loop is to execute the body as long as the condition φ evaluates to true.

Note that these programs are not guaranteed to halt. For those input databases it does not, the query represented by the program is not defined on that particular input database.

3.2.2 Generic Spatial Database Queries

We defined a constraint database query as a computable mapping. But, in database theory, it is common to also ask that queries are *generic* (see Definition 2.4). A generic query asks only for properties that are shared by "isomorphic" encodings of

the same data or, in other words, the result of a generic query depends only to a certain, limited extent on the actual internal representation of the database it is applied to. Chandra and Harel [11] considered the permutations of the universal domain $\mathcal{U}$ of the database (that possibly fix some elements of the domain) as "isomorphisms" for relational databases. Paredaens, Van den Bussche and Van Gucht [58] have shown that for spatial databases, the definition of genericity is not unique and that it depends on the particular kind of geometry in which the spatial information is to be interpreted. Genericity of spatial databases hence is defined as a function of some group of geometric transformations.

The following taxonomy of genericity classes was proposed, motivated by spatial database practice. This taxonomy is indexed by the chain of transformation groups of $\mathbb{R}^n$:

$$\mathcal{T} \subset \mathcal{I} \subset \mathcal{D} \subset \mathcal{S} \subset \mathcal{A} \subset \mathcal{H} \subset \mathcal{P},$$

where $\mathcal{T}$ is the group of the translations, $\mathcal{I}$ the group of the isometries, $\mathcal{D}$ the group of the direct isometries, $\mathcal{S}$ the group of similarities, $\mathcal{A}$ the group of the affinities and $\mathcal{H}$ the group of the homeomorphisms. Finally, $\mathcal{P}$ is the group of all permutations of $\mathbb{R}^n$. Clearly, if $\mathcal{G}'$ is a subgroup of $\mathcal{G}$, all $\mathcal{G}$-generic queries are also $\mathcal{G}'$-generic.

An important result regarding genericity of constraint database queries, is the following [43]. Let $\mathcal{G}$ be one of the transformation groups introduced in the previous paragraph. Then $\mathcal{G}$-genericity is an undecidable property of FO(+, ×, <, 0, 1)-queries.

Gyssens, Van den Bussche and Van Gucht [43, 50] showed, however, that the $\mathcal{G}$-generic FO(+, ×, <, 0, 1)-queries are recursively enumerable, for $\mathcal{G}$ in the chain $\mathcal{T} \subset \mathcal{I} \subset \mathcal{D} \subset \mathcal{S} \subset \mathcal{A} \subset \mathcal{H}$. For each $\mathcal{G}$ in this chain, they defined a first-order point language FO($\Pi_{\mathcal{G}}$), parameterized by a set $\Pi_{\mathcal{G}}$ of point predicates. These languages are sound and complete for the corresponding $\mathcal{G}$-generic FO(+, ×, <, 0, 1)-queries.

The point predicates are:

(i) **Between**$(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$, which means that the n-dimensional point $\mathbf{a}_2$ lies on the closed line segment between the n-dimensional points $\mathbf{a}_1$ and $\mathbf{a}_3$;

(ii) **EqDist**$(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4)$, which means that the (Euclidean) distance between $\mathbf{a}_1$ and $\mathbf{a}_2$ equals the distance between $\mathbf{a}_3$ and $\mathbf{a}_4$;

(iii) **UnitDist**$(\mathbf{a}_1, \mathbf{a}_2)$, which means that the distance between $\mathbf{a}_1$ and $\mathbf{a}_2$ equals one;

(iv) **Pos**$(\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_n)$, which means that $(\mathbf{a}_0, \mathbf{a}_1, \ldots, \mathbf{a}_n)$ is a positively oriented basis of $\mathbb{R}^n$; and

(v) **Smaller**$_i(\mathbf{a}_1, \mathbf{a}_2)$, which means that the i-th coordinate of $\mathbf{a}_1$ is less or equal to the i-th coordinate of $\mathbf{a}_2$.

The next table shows for each transformation group $\mathcal{G}$, the corresponding set $\Pi_{\mathcal{G}}$ of point predicates.

$\mathcal{G}$	Point predicate set $\Pi_{\mathcal{G}}$
$\mathcal{A}$	{**Between**}
$\mathcal{S}$	{**Between**, **EqDist**}
$\mathcal{I}$	{**Between**, **EqDist**, **UnitDist**}
$\mathcal{D}$	{**Between**, **EqDist**, **UnitDist**, **Pos**}
$\mathcal{T}$	{**Between**, **EqDist**, **UnitDist**, **Pos**, **Smaller**$_i (1 \leq i \leq n)$}

3.3 Spatio-temporal Constraint Databases

The constraint model is studied in the context of spatial databases, but it provides an equally elegant and efficient way to model spatio-temporal data. Any spatio-temporal object that is a moving n-dimensional spatial object, can be seen as a spatial $(n+1)$-dimensional object. Figure 3.1 for example, can be described as a spatio-temporal relation showing a point, followed by a growing and shrinking disk. Afterwards, it shows a disk that grows, shrinks, grows again and finally shrinks into a point. But, one could also describe the same figure as a $(2+1)$-dimensional object showing the union of a filled sphere and the boundary of two intersecting spheres. We now define spatio-temporal databases, as a special case of constraint databases.

3.3.1 Spatio-temporal Databases as Constraint Databases

The underlying domain of a spatio-temporal database is $(\mathbb{R}^n \times \mathbb{R})$. We prefer the notation $(\mathbb{R}^n \times \mathbb{R})$ over $\mathbb{R}^{n+1}$ for the domain because it stresses the distinction between the time coordinate and the n spatial coordinates of the $(n+1)$-dimensional points. We will also call n the *underlying dimension* of a spatio-temporal database. For the remainder of this text, we assume, for technical reasons that will become clear in Section 4.2, that $n \geq 2$.

Throughout this thesis we will often use the canonical bijection

$$can_{ST} : (\mathbb{R}^n \times \mathbb{R})^k \to \mathbb{R}^{(n+1)\times k}$$

that maps a tuple $((\mathbf{a}_1, \tau_1), \ldots, (\mathbf{a}_k, \tau_k))$ to $(a_{1,1}, \ldots, a_{1,n}, \tau_1, \ldots, a_{k,1}, \ldots, a_{k,n}, \tau_k)$, where for $1 \leq i \leq k$ and $1 \leq j \leq n$, $a_{i,j}$ denotes the jth real coordinate of $\mathbf{a}_i$.

Definition 3.10 (**Spatio-temporal database**)**.** A *(spatio-temporal) database schema* σ^{st} is a finite set of relation names, where each relation name R^{st} has a natural number $ar(R^{st})$, called its arity, assigned to it.

A subset of $(\mathbb{R}^n \times \mathbb{R})^k$ is a *spatio-temporal relation of arity* k if its image under the canonical bijection $can_{ST} : (\mathbb{R}^n \times \mathbb{R})^k \to \mathbb{R}^{(n+1)\times k}$ is a constraint relation of arity $(n+1) \times k$.

Let σ^{st} be a (spatio-temporal) database schema. A *spatio-temporal database over* σ^{st} is a mapping $\mathfrak{F}$ on σ^{st} assigning to each relation name R^{st} in σ^{st}, a spatio-temporal relation $R^{st^{\mathfrak{F}}}$ of arity k, where $k = ar(R^{st})$.

Remark 3.11. A spatio-temporal database $\mathfrak{F}$ over σ^{st} can be viewed in a natural way as a constraint database D over the constraint schema σ, which has for each

relation name R^{st} of σ^{st}, a relation name R of arity $(n+1) \times ar(R^{st})$. For each relation name R^{st}, R^D is obtained from $R^{st^{\mathfrak{F}}}$ by applying the canonical bijection $can_{ST} : (\mathbb{R}^n \times \mathbb{R})^{ar(R)} \to \mathbb{R}^{(n+1)\times ar(R)}$. We will use the notation introduced here, throughout this thesis.

Following this remark, we observe that spatio-temporal relations and databases can be *finitely* encoded and stored by means of the systems of polynomial equalities and inequalities, *i.e.*, by means of a quantifier-free formula of first-order logic over the reals with $+$, $\times$, $<$ and the constants 0 and 1, that describe the associated constraint relations and databases.

We also remark the following.

Remark 3.12. The data model presented here and the query languages presented in this thesis can be extended straightforwardly to the situation where spatio-temporal relations are accompanied by classical thematic information. However, because the problem that is discussed here is captured by this simplified model, we stick to it for reasons of simplicity of exposition.

Example 3.13. Figure 3.1 gives an illustration of a spatio-temporal database over a schema $\sigma^{st} = \{R^{st}\}$ with underlying dimension 2, where R^{st} has arity 1. It shows at its beginning, *i.e.*, at $t = -1$, a single point in the origin of $\mathbb{R}^2$. Then it shows a disk whose radius increases and later decreases and ends in a point at moment $t = 1$, followed by a circle whose radius increases, decreases, increases and then shrinks to a point. □

Definition 3.14 (**Snapshot**)**.** Let σ^{st} be a spatio-temporal schema and let $\mathfrak{F}$ be a spatio-temporal database over σ^{st} with underlying dimension n. Let R^{st} be a relation name in σ^{st} and let τ_0 be a real number representing a moment in time. We call the subset

$$R^{st^{\mathfrak{F}}} \cap (\mathbb{R}^n \times \{\tau_0\})^{ar(R^{st})}$$

of $(\mathbb{R}^n \times \{\tau_0\})^{ar(R)}$ the *snapshot of* R^{st} *at the moment* τ_0. The snapshot of the spatio-temporal database $\mathfrak{F}$ at the moment τ_0 is the finite set of snapshots of all its relations at τ_0.

Example 3.15. For the spatio-temporal relation depicted in Figure 3.1, the snapshot at -1 is $\{(0, 0, -1)\}$, the snapshot at 0 is the closed unit disk in the plane $t = 0$ and the snapshot at 5 is the empty set. □

3.3.2 Spatio-temporal Constraint Database Queries

We now define spatio-temporal database queries, and show how FO($+$, $\times$, $<$, 0, 1) can be used as a spatio-temporal database query language.

Definition 3.16 (**Spatio-temporal database query**)**.** Let σ^{st} be a spatio-temporal database schema and let us consider input spatio-temporal databases over σ^{st} with underlying dimension n. A *k-ary n-dimensional spatio-temporal database query Q over σ^{st}* is a partial, computable mapping (in the sense of Remark 3.5) from the set of spatio-temporal databases over σ^{st} to the set of k-ary spatio-temporal relations with underlying dimension n.

We also call a k-ary n-dimensional spatio-temporal database query a *spatio-temporal database query of output type* (n, k).

Note that we restrict spatio-temporal database queries to preserve the underlying dimension of the input database.

Example 3.17. Let $\sigma^{st} = \{R^{st}\}$, where R^{st} has arity 1 and let the underlying dimension be 2. The query that selects those snapshots from the relation R^{st} where R^{st} shows a circle is a spatio-temporal database query of output type $(2, 1)$. Applied to the database of Example 3.13 and shown in Figure 3.1, this query returns the union of its snapshots in the open time interval $]1, 4[$. □

There is a natural way to see spatio-temporal queries as constraint queries, that is captured in the following definition of equivalence of queries. Here, $canDB(\mathfrak{F})$ is the database instance consisting of the relation instances $can_{ST}(R_i^{st^{\mathfrak{F}}})$, for all relation names R_i^t of σ^{st}.

Definition 3.18 (**Spatio-temporal queries and constraint queries**)**.** Let $\sigma^{st} = \{R_1^{st}, R_2^{st}, \ldots, R_m^{st}\}$ be a spatio-temporal database schema and let us consider input spatio-temporal databases over σ^{st} with underlying dimension n. Let σ be the corresponding constraint database schema (see Remark 3.11). Let Q be a k-ary n-dimensional spatio-temporal database query over σ^{st} and let Q' be a $((n+1)\times k)$-ary constraint database query over σ. We say that Q and Q' are *equivalent* if for every database $\mathfrak{F}$ over σ^{st} we have

$$can_{ST}(Q(\mathfrak{F})) = Q'(canDB(\mathfrak{F})).$$

We introduce FO(+, ×, <, 0, 1) here as a spatio-temporal query language, albeit on constraint databases that represent spatio-temporal databases.

Definition 3.19 (**First-order logic as a spatio-temporal query language**)**.** Let $\sigma^{st} = \{R_1^{st}, R_2^{st}, \ldots, R_m^{st}\}$ be a spatio-temporal database schema and let us consider queries working on input databases over σ^{st} with underlying dimension n. Let R_i, $1 \leq i \leq m$, be the corresponding constraint relation names of arity $(n+1)\times ar(R_i^{st})$ (we follow the notation of Remark 3.11) and let σ be the constraint schema $\{R_1, R_2, \ldots, R_m\}$.

Let $\varphi(\mathbf{x}_1, t_1, \mathbf{x}_2, t_2, \ldots, \mathbf{x}_k, t_k)$, be a first-order logic formula over the alphabet (+, ×, <, 0, 1, $R_1, R_2, \ldots, R_m$). If $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n})$, then the free variables of φ are $x_{1,1}, \ldots, x_{1,n}, t_1, x_{2,1}, \ldots, x_{2,n}, t_2, \ldots, x_{k,1}, \ldots, x_{k,n}, t_k$. The formula φ expresses a constraint $((n+1)\times k)$-ary query Q' which is equivalent to a k-ary n-dimensional spatio-temporal query Q. For each input spatio-temporal database $\mathfrak{F}$ over σ, $Q(\mathfrak{F})$ is defined as the set of points $((\mathbf{a}_1, \tau_1), (\mathbf{a}_2, \tau_2), \ldots, (\mathbf{a}_k, \tau_k))$ of $(\mathbb{R}^n \times \mathbb{R})^k$ such that

$$(\mathbb{R}, +, \times, 0, 1, <, R_1^{canDB(\mathfrak{F})}, R_2^{canDB(\mathfrak{F})}, \ldots, R_m^{canDB(\mathfrak{F})}) \models \varphi[\mathbf{a}_1, \tau_1, \mathbf{a}_2, \tau_2, \ldots, \mathbf{a}_k, \tau_k],$$

where $\varphi[\mathbf{a}_1, \tau_1, \mathbf{a}_2, \tau_2, \ldots, \mathbf{a}_k, \tau_k]$ denotes the formula $\varphi(\mathbf{x}_1, t_1, \mathbf{x}_2, t_2, \ldots, \mathbf{x}_k, t_k)$ with its free variables instantiated by $\mathbf{a}_1, \tau_1, \mathbf{a}_2, \tau_2, \ldots, \mathbf{a}_k, \tau_k$.

Example 3.20. As in Example 3.17, let $\sigma^{st} = \{R^{st}\}$, where R^{st} has arity 1 and let the underlying dimension be 2. The formula

$$\exists x_0\, \exists y_0\, \exists r\, (r > 0) \wedge ((x - x_0)^2 + (y - y_0)^2 = r^2 \leftrightarrow R(x, y, t))$$

expresses a spatio-temporal query of output type $(2, 1)$. It selects those snapshots from a spatio-temporal relation R^{st} where R^{st} shows a circle. As mentioned, applied to the database of Example 3.13, this query returns all its snapshots in the time interval $]1, 4[$. □

Remark 3.21. An arbitrary FO(+, ×, <, 0, 1)-formula does not necessarily express a spatio-temporal database query as shown by the following example. The formula

$$\exists t\,(R(x_1, x_2, t))$$

expresses the projection of the spatio-temporal relation R^{st} on the spatial (x_1, x_2)-plane. The formula returns a set of couples (x_1, x_2) in $\mathbb{R}^2$ that form a semi-algebraic set with a purely spatial meaning.

In the same way as FO(+, ×, <, 0, 1), also the language FO(+, ×, <, 0, 1)+While can be used as a spatio-temporal query language.

4

Generic Spatio-temporal Query Languages

In database theory, it is usually required that queries are generic. On a technical level, generic queries are defined as being invariant under those transformations of the data that preserve the relevant aspects of the data. Whereas Chandra and Harel [11] considered the group of the isomorphisms (that possibly fix some elements of the domain) in the case of relational databases (see also Definition 2.4), Paredaens, Van den Bussche and Van Gucht [43] identified different geometrical and topological transformation groups for spatial database applications (see Section 3.2.2).

We investigate which notions of genericity are appropriate for spatio-temporal databases and which transformation groups express them. We observe that the transformations should first and foremost respect the monotone and unidirectional nature of time, *i.e.*, leave the temporal order of events unchanged. It follows that the relevant transformation groups are the product of a group of time-(in)dependent spatial transformations and a group of monotone increasing transformations of the time-component of the spatio-temporal data. Next, we focus on the former groups and study which of them leave different spatial and spatio-temporal properties (like collinearity, distance and orientation) unchanged. We also focus on physical properties of spatio-temporal data (like velocity and acceleration). The transformation groups that we consider are all subgroups of the time-dependent or time-independent affinities of $(\mathbb{R}^n \times \mathbb{R})$.

We study the notion of spatio-temporal genericity relative to two popular query languages in the constraint model: first-order logic over the reals (FO(+, $\times$, $<$, 0, 1), see Definition 3.6) and an extension of this logic with a while-loop (FO(+, $\times$, $<$, 0, 1)+While, see Definition 3.9). First, we show that all the genericity classes are undecidable. We show that the considered classes of generic first-order queries are

recursively enumerable, however. Hereto, we define first-order point-based languages in which variables are assumed to range over points in $(\mathbb{R}^n \times \mathbb{R})$ and which contain certain point predicates (such as $\mathbf{Between}^{((n+1))}$ and **Before**). These point-based languages are shown to be sound and complete for the first-order queries in the considered genericity classes. We have also shown that extensions of these point-based logics with a While-loop give sound and complete languages for the computable queries in the different genericity classes. Our results are inspired by similar results that were obtained by Gyssens, Van den Bussche and Van Gucht in the context of spatial databases [43, 71]. Also, the proof techniques we use for time-independent transformation groups, are generalizations of techniques introduced in those papers. However, our results for genericity notions described by time-dependent transformations require new proof techniques.

This chapter is organized as follows. In Section 4.1, we investigate what genericity means in the context of spatio-temporal databases. In Section 4.2, we present sound and complete first-order query languages for the different genericity classes that we identified in Section 4.1. In Section 4.3, we present languages for expressing all computable generic queries, for the proposed genericity classes.

4.1 Spatio-temporal Genericity

As stated in the introduction, we are interested in spatio-temporal database queries that are invariant under the elements of a certain spatio-temporal transformation group (for function composition)

$$\mathcal{F} = \{f \mid f = (f_1, f_2, \ldots, f_n, f_t) : (\mathbb{R}^n \times \mathbb{R}) \to (\mathbb{R}^n \times \mathbb{R})\}.$$

The idea is that the result of spatio-temporal queries should be largely independent of the particular coordinate system in which the data are presented. In this section, we formalize this idea by the notion of $\mathcal{F}$-genericity.

In the remainder of this section, we look at different types of transformation groups and we impose two further conditions on these transformations. Firstly, we look at purely temporal conditions. Secondly, we look at purely spatial or spatio-temporal conditions that reflect the nature of the queries one is interested in. We also look at transformation groups that are suited for applications in which physical notions such as velocity and acceleration are of importance.

4.1.1 Spatio-temporal Genericity

Let $f : (\mathbb{R}^n \times \mathbb{R}) \to (\mathbb{R}^n \times \mathbb{R})$ be a function, let R^{st} be a spatio-temporal relation name of arity k and let $R^{st^{\mathfrak{F}}}$ be a spatio-temporal relation instance with underlying dimension n. In the following, we use the notation $f(R^{st^{\mathfrak{F}}})$ to abbreviate the set

$$\{(f(\mathbf{a}_1, \tau_1), f(\mathbf{a}_2, \tau_2), \ldots, f(\mathbf{a}_k, \tau_k)) \in (\mathbb{R}^n \times \mathbb{R})^k \mid (\mathbf{a}_1, \tau_1, \mathbf{a}_2, \tau_2, \ldots, \mathbf{a}_k, \tau_k) \in R^{st^{\mathfrak{F}}}\}.$$

Definition 4.1 (**Spatio-temporal genericity**)**.** Let Q be a spatio-temporal database query that takes databases of signature $\sigma^{st} = \{R_1^{st}, \ldots, R_m^{st}\}$ as input, with

underlying dimension n. Let $\mathcal{F} = \{f \mid f : (\mathbb{R}^n \times \mathbb{R}) \to (\mathbb{R}^n \times \mathbb{R})\}$ be a spatio-temporal transformation group. We say that Q is $\mathcal{F}$-*generic* if and only if, for any f in $\mathcal{F}$ and for each pair of spatio-temporal databases $\mathfrak{F}_1$ and $\mathfrak{F}_2$ over σ^{st},

$$\mathfrak{F}_2 = (R_1^{st\mathfrak{F}_2}, \ldots, R_m^{st\mathfrak{F}_2}) = (f(R_1^{st\mathfrak{F}_1}), \ldots, f(R_m^{st\mathfrak{F}_1}))$$

implies that $f(Q(\mathfrak{F}_1)) = Q(\mathfrak{F}_2)$.

This definition will be illustrated in Section 4.1.5. It is clear that if a query is $\mathcal{F}$-generic, it is also $\mathcal{F}'$-generic for any subgroup $\mathcal{F}'$ of $\mathcal{F}$.

4.1.2 Temporal Restrictions

It is very natural to describe spatio-temporal events with the notions "before", "after" and "co-temporal". For instance, when two people arrive shortly after each other, we say *"Mary arrived before Jane"* rather than *"Mary arrived at 9:31 and Jane at 9:35"*. Another example is any kind of race. The winner is the one that finishes first. So, foremost the order of arrival of the participants matters. Exact time moments are only important in very specific situations.

We start with the definition of a spatio-temporal event.

Definition 4.2 (**Event**)**.** An *event* is a subset of $(\mathbb{R}^n \times \mathbb{R})$. The projection of an event A on the time-axis is denoted by $\pi_t(A)$ and called the *time-domain* of A.

Let A and B be events. In the terminology of Allen's interval calculus [5, 6], A and B are called *co-temporal* if $\pi_t(A) = \pi_t(B)$ (we denote this by $A =_t B$). Allen says A is *before* B if $t_A < t_B$ for all $t_A \in \pi_t(A)$ and all $t_B \in \pi_t(B)$ (we denote this by $A <_t B$).

Remark that $A \leq_t B := (A =_t B$ or $A <_t B)$ is a pre-order on events.

Definition 4.3 (**Temporal order-preserving transformation**)**.** We say that a transformation $f : (\mathbb{R}^n \times \mathbb{R}) \to (\mathbb{R}^n \times \mathbb{R})$ *preserves the order of events* if for all events A and B, $A =_t B$ implies $f(A) =_t f(B)$ and $A <_t B$ implies $f(A) <_t f(B)$.

Proposition 4.4. A transformation $f = (f_1, f_2, \ldots, f_n, f_t) : (\mathbb{R}^n \times \mathbb{R}) \to (\mathbb{R}^n \times \mathbb{R}) : (\mathbf{x}, t) \mapsto (f_1(\mathbf{x}, t), \ldots, f_n(\mathbf{x}, t), f_t(\mathbf{x}, t))$ preserves the order of events if and only if f_t is a strictly monotone increasing bijection of t alone.

Proof. The if-direction is straightforward. To prove the other direction, let $f = (f_1, f_2, \ldots, f_n, f_t)$ be a transformation of $(\mathbb{R}^n \times \mathbb{R})$. Consider any two events $A = \{(a_1, a_2, \ldots, a_n, \tau)\}$ and $B = \{(a'_1, a'_2, \ldots, a'_n, \tau)\}$. Since $A =_t B$, then $f_t(a_1, a_2, \ldots, a_n, \tau) = f_t(a'_1, a'_2, \ldots, a'_n, \tau)$. This shows that f_t is a function of t alone.

Consider any two events $A = \{(a_1, a_2, \ldots, a_n, \tau_1)\}$ and $B = \{(a_1, a_2, \ldots, a_n, \tau_2)\}$ with $\tau_1 < \tau_2$. Since $A <_t B$, then $f_t(\tau_A) < f_t(\tau_B)$. This shows that f_t is a strictly monotone function of t.

The transformation groups that we consider are all groups with respect to the composition operator $\circ$ of functions. Therefore, for every transformation f also its inverse exists, and hence f is a bijection. Given the fact that the component f_t is a function of t alone, it has to be a bijection too. □

We require that transformations preserve the order of events. We can therefore write the transformation groups of interest as a product of groups, *i.e.*, $\mathcal{F} = (\mathcal{F}_{st}, \mathcal{F}_t)$, where

$$(\mathcal{F}_{st}, \mathcal{F}_t) = \{(f_{st}, f_t) \mid f_{st} = (f_1, f_2, \ldots, f_n) : (\mathbb{R}^n \times \mathbb{R}) \to \mathbb{R}^n \text{ and } f_t : \mathbb{R} \to \mathbb{R}\}.$$

The particular groups $\mathcal{F}_t$ that we will consider in this thesis are:

- $\mathcal{A}_t = \{t \mapsto at + b \mid a, b \in \mathbb{R} \text{ and } a > 0\}$, *i.e.*, the monotone affinities of the time-line;
- $\mathcal{T}_t = \{t \mapsto t + b \mid b \in \mathbb{R}\}$, *i.e.*, the translations of the time-line; and
- $Id_t = \{\text{id}\}$, *i.e.*, the identity of time.

Invariance with respect to these types of transformations of time is often encountered in physics [20].

4.1.3 Spatial and Spatio-temporal Restrictions

In the following, we consider transformations coming from practical situations where (i) moving objects are monitored from a fixed position, where (ii) a fixed object is observed from a moving position or where (iii) a moving object is observed from a moving position. The frame of reference is therefore changing in a time-dependent way. In real life, this continuous change of reference system arises in different kinds of situations. For example, when a person in a helicopter and another person on a motorbike both film a race, the movie they make of the race will be related to their position and orientation at each time moment.

The general form of the transformation groups $\mathcal{F}_{st}$ that we consider have elements of the form:

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ t \end{pmatrix} \mapsto \begin{pmatrix} \alpha_{11}(t) & \alpha_{12}(t) & \cdots & \alpha_{1n}(t) \\ \alpha_{21}(t) & \alpha_{22}(t) & \cdots & \alpha_{2n}(t) \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_{n1}(t) & \alpha_{n2}(t) & \cdots & \alpha_{nn}(t) \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \beta_1(t) \\ \beta_2(t) \\ \vdots \\ \beta_n(t) \end{pmatrix},$$

where the α_{ij} and β_i are functions from $\mathbb{R}$ to $\mathbb{R}$. Furthermore, we require that the transformation groups that we consider are semi-algebraic (we give a precise definition of semi-algebraic transformation groups in Section 4.3.3).

We will consider the following groups $\mathcal{F}_{st}$ of transformations:

- $\mathcal{A}_{st}$ is the group of transformations of the above form where the $\alpha_{ij}(t)$ and $\beta_i(t)$ are arbitrary functions of t such that the matrix of the $\alpha_{ij}(t)$ has an inverse for each value of t, *i.e.*, these are the *time-dependent affinities*;
- $\mathcal{A}^f_{st}$ is the subgroup of $\mathcal{A}_{st}$ consisting of transformations for which the functions $\alpha_{ij}(t)$ and $\beta_i(t)$ only take a finite number of values, *i.e.*, functions that are piecewise constant;

- $\mathcal{A}^c_{st}$ is the subgroup of $\mathcal{A}^f_{st}$ consisting of transformations for which the functions $\alpha_{ij}(t)$ are constants and $\beta_i(t)$ are linear functions of t;
- $\mathcal{S}_{st}$, $\mathcal{S}^f_{st}$ and $\mathcal{S}^c_{st}$ are subgroups of $\mathcal{A}_{st}$, $\mathcal{A}^f_{st}$ and $\mathcal{A}^c_{st}$ respectively, where the matrix of the $\alpha_{ij}(t)$ represents at each moment a similarity, *i.e.*the composition of an isometry (given by a matrix with determinant 1) and a scaling (given by a non-zero multiple of the unit matrix);
- $\mathcal{I}_{st}$, $\mathcal{I}^f_{st}$, $\mathcal{I}^c_{st}$ are the subgroups of the above groups where the determinant of the matrix consisting of the $\alpha_{ij}(t)$ equals 1 at each moment, *i.e.*, this matrix determines an isometry;
- $\mathcal{T}_{st}$, $\mathcal{T}^f_{st}$, $\mathcal{T}^c_{st}$ are the subgroups of the above groups where the matrix consisting of the $\alpha_{ij}(t)$ is the identity matrix, *i.e.*, these are groups of translations.

4.1.4 Physical Transformation Groups

The following groups are of interest when notions such as velocity, acceleration and force are important in an application. These transformation groups can be found by solving the differential equations that express that these physical entities are preserved [20]. We consider these notions for arbitrary and rigid motions, respectively. A *rigid motion* is a motion that preserve the shape of a moving body or moving figure, *i.e.*, it is an isometric movement. To study the velocity and acceleration of a moving body, we only consider the movement of the center of mass of that figure and do not take into account the changes in shape of the body.

The transformation groups of interest here are the following.

- $\mathcal{V}_{st}$ is the subgroup of $\mathcal{A}^c_{st}$ where the β_i are constants. This group of transformations preserves the *velocity vector* of a moving figure.
- $\mathcal{V}(\mathcal{R})_{st}$ is the subgroup of $\mathcal{I}^c_{st}$ where the β_i are constants. This group of transformations preserves the *velocity vector* of a moving figure in rigid motion.
- $\mathcal{AC}_{st}$ is the group $\mathcal{A}^c_{st}$. This group of transformations preserves the *acceleration vector* of a moving object.
- $\mathcal{AC}(\mathcal{R})_{st}$ is the group $\mathcal{I}^c_{st}$. This group of transformations preserves the *acceleration vector* of a moving figure in rigid motion.

In physics it is customary to consider only translations for what concerns the time dimension, *i.e.*, the transformations in the group $\mathcal{T}_t$. The group $(\mathcal{AC}(\mathcal{R})_{st}, \mathcal{T}_t)$ is also known as *the group of the Galilei transformations* [20]. It is particularly useful because all laws of classical mechanics are invariant for this group of transformations of space-time [20].

4.1.5 Examples of Generic Queries

We end this section with a number of examples of queries that are generic for some of the genericity classes that we have introduced above.

Suppose in some city, an experiment is set up to evaluate the traffic situation. A number of probe cars (for simplicity, we assume two) is continuously driving around the city in a random way. The trajectories of the cars are stored in a spatio-temporal database, of underlying dimension 2, over the schema $\sigma^{st} = \{carA, carB\}$, where the spatio-temporal relations $carA$ and $carB$ both have arity 1. For those queries that are first-order-expressible, we give the formulas expressing them. The constraint relation names corresponding to the spatio-temporal relation names $carA$ and $carB$ are A and B, respectively. In these examples, we assume that time is measured in seconds and distance is measured in meters. We indicate for each example query the most general transformation group it is generic for.

Example 4.5. Q_1 : *Does the route followed by car A self-intersect more often than the route followed by car B does?*

This query is $(\mathcal{V}_{st}, \mathcal{A}_t)$-generic, but not $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic, for instance. It is not expressible in first-order logic. In Section 4.3, we will give a "program" expressing this query. □

Example 4.6. Q_2 : *Give the places and time moments where it is true for car A that when it reaches them, it is standing still at that spot for at least* 300 *more seconds, (*i.e.*, where and when did car A encounter a traffic jam?).*

This query is $(\mathcal{V}_{st}, \mathcal{T}_t)$-generic. Indeed, the fact that a car has speed zero (when it is standing still) has to be preserved, which requires the group $\mathcal{V}_{st}$, and the length of time intervals has to be preserved, which requires $\mathcal{T}_t$. This query is expressed by the following FO(+, ×, <, 0, 1, A)-formula:

$$\varphi_2(x, y, t) := (A(x, y, t) \wedge \forall t_2 \, ((t \leq t_2 \wedge t_2 \leq t + 300) \rightarrow A(x, y, t_2)).$$

□

Example 4.7. Q_3 : *Was there a collision between car A and car B?*

This query is $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic. This query is expressed by the following FO(+, ×, <, 0, 1, A, B)-formula:

$$\varphi_3 := \exists x \, \exists y \, \exists t \, (A(x, y, t) \wedge B(x, y, t)).$$

□

Example 4.8. Q_4 : *Did car A pass at 500 meters north of car B at time moment* $t = 5930$*?*

This query is $(\mathcal{T}_{st}, Id_t)$-generic. This query is expressed by the following FO(+, ×, <, 0, 1, A, B)-formula:

$$\varphi_4 := \exists x_1 \, \exists y_1 \, \exists y_2 \, (A(x_1, y_1, 5930) \wedge B(x_1, y_2, 5930) \wedge y_1 = y_2 + 500).$$

□

Example 4.9. *Q_5 : Did car A encounter any "empty roads"? (I.e., were there parts of its trajectory where it could drive at constant speed in a straight line for at least 6000 seconds?)*

This query is $(\mathcal{AC}_{st}, \mathcal{T}_t)$-generic. The fact that a car drives at constant speed (*i.e.*, has an acceleration of zero) has to be preserved. Note that, because the car's movement is a polynomial function of time, driving at constant speed means driving in a straight line. Query Q_5 can be expressed by the following FO(+, ×, <, 0, 1, A)-formula:

$$\varphi_5 := \exists t_1\, \exists t_2\, \exists x_1\, \exists y_1\, \exists x_2\, \exists y_2\, (A(x_1, y_1, t_1) \wedge A(x_2, y_2, t_2) \wedge$$
$$t_2 = t_1 + 6000 \wedge \forall t_3\, ((t_1 \leq t_3 \wedge t_3 \leq t_2) \rightarrow \exists x_3\, \exists y_3\, (A(x_3, y_3, t_3) \wedge$$
$$(t_2 - t_1)x_3 = (t_2 - t_3)x_1 + (t_3 - t_1)x_2 \wedge (t_2 - t_1)y_3 = (t_2 - t_3)y_1 + (t_3 - t_1)y_2))).$$

□

This completes the examples section. We will go back to these examples later on, when we have defined point languages.

4.2 First-order Generic Spatio-temporal Queries

In this section, we study the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic queries that are expressible in FO(+, ×, <, 0, 1, σ^{st}). To start with, we give a general undecidability result. We prove that it is undecidable whether a query is $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic, for any nontrivial group $(\mathcal{F}_{st}, \mathcal{F}_t)$.

Next, we show that $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic FO(+, ×, <, 0, 1, σ^{st})-queries are recursive enumerable, however. We do this by syntactically specifying languages that capture the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic queries, for all groups $(\mathcal{F}_{st}, \mathcal{F}_t)$ listed in Section 4.1.3 and Section 4.1.4.

The strategy to prove the following Theorem was introduced by Paredaens, Van den Bussche and Van Gucht [58].

Theorem 4.10 (**$(\mathcal{F}_{st}, \mathcal{F}_t)$-genericity is undecidable**). For all non-trivial groups $(\mathcal{F}_{st}, \mathcal{F}_t)$ mentioned in the previous section, $(\mathcal{F}_{st}, \mathcal{F}_t)$-genericity of spatio-temporal FO(+, ×, <, 0, 1)σ^{st}-queries is undecidable, where σ^{st} is a non-empty spatio-temporal database schema.

Proof. Let $\mathcal{F}$ be a group of transformations of $(\mathbb{R}^n \times \mathbb{R})$ that contains an element f_0 that does not map $(0, 0)$ to itself. We show that $\mathcal{F}$-genericity of spatio-temporal queries over a certain schema $\sigma^{st} = \{R^{st}\}$, where R^{st} is a one-dimensional unary spatio-temporal relation, of output type (1,1) is undecidable. For other non-empty schemas the proof is similar. We will do this by reducing deciding the truth of sentences of the $\forall^*$-fragment of number theory to the genericity question. The $\forall^*$-fragment of number theory is known to be undecidable since Hilbert's 10th problem [54] can be formulated in it.

We encode a natural number n by the unary one-dimensional spatio-temporal relation

$$enc(n) := \{(0,0),(1,0),\ldots,(n,0)\}.$$

A (k-dimensional) vector of natural numbers $(n_1, n_2, \ldots, n_k)$ is encoded by the relation

$$enc(n_1, n_2, \ldots, n_k) := enc(n_1) \cup (enc(n_2) + (n_1 + 2, 0)) \cup$$

$$\ldots \cup (enc(n_k) + (n_1 + 2 + \cdots + n_{k-1} + 2, 0)).$$

For fixed k, the corresponding decoding is expressible in FO($+$, $\times$, $<$, 0, 1). We thus associate to the first-order sentence $\forall n_1 \cdots \forall n_k\, \varphi(n_1, \ldots, n_k)$ of number theory the following spatio-temporal query Q_φ over the input schema $\sigma^{st} = \{R^{st}\}$:

---

if R^{st} encodes a vector $(n_1, \ldots, n_k) \in \mathbb{N}^k$ **then**
  if $\varphi(n_1, \ldots, n_k)$ **then**
    return $\emptyset$
  else
    return $\{(0,0)\}$
  end if
else
  return $\emptyset$.
end if

This query is expressible in FO($+$, $\times$, $<$, 0, 1, $\{R\}$), where R is the constraint relation corresponding to R^{st}.

Claim. The query Q_φ is $\mathcal{F}$-generic if and only if the sentence $\forall n_1 \cdots \forall n_k\, \varphi(n_1, \ldots, n_k)$ holds in the natural numbers.

Now, we prove this claim. First, suppose that, for all $(n_1, \ldots, n_k) \in \mathbb{N}^k$, $\varphi(n_1, \ldots, n_k)$ holds. Let R^{st} be a one-dimensional unary spatio-temporal relation and let f be some transformation of $\mathcal{F}$. We have to prove that

$$f(Q_\varphi(R^{st})) = Q_\varphi(f(R^{st})).$$

The result of $Q_\varphi(R^{st})$ will always be $\emptyset$: either R^{st} does not encode a vector $(n_1, \ldots, n_k)$, or it does and $\varphi(n_1, \ldots, n_k)$ holds. For the same reason, $Q_\varphi(f(R^{st}))$ also equals $\emptyset$. The transformation f maps $\emptyset$ to $\emptyset$, hence $f(Q_\varphi(R^{st})) = \emptyset$, which concludes the first part of the proof.

Now assume that there exists an $(n_{0,1}, \ldots, n_{0,k})$ such that $\varphi(n_{0,1}, \ldots, n_{0,k})$ is not true. Let R^{st} be the relation that decodes $(n_{0,1}, \ldots, n_{0,k})$. The result of $Q_\varphi(R^{st})$ will be the origin $(0,0)$ of $\mathbb{R} \times \mathbb{R}$. If we now apply f_0 to this result, the output is a vector $(y, z) \neq (0,0)$. On the other side, if we first apply f_0 to R^{st}, there are three possibilities. Either $f_0(R^{st})$ encodes a vector $(n1, 1, \ldots, n_{1,k})$ for which $\varphi(n_{1,1}, \ldots, n_{1,k})$ is true, then the result of $Q_\varphi(f_0(R^{st}))$ will be $\emptyset$. Or, $f_0(R^{st})$ encodes a vector $(n_{1,1}, \ldots, n_{1,k})$ for which $\varphi(n_{1,1}, \ldots, n_{1,k})$ is not true, and $Q_\varphi(f_0(R^{st}))$ returns $(0,0)$. In the last case, $f_0(R^{st})$ does not encode a vector of natural numbers, in

which case the result of $Q_\varphi(f_0(R^{st}))$ will be $\emptyset$ again. In all cases, we have that $Q_\varphi(f_0(R^{st})) \neq f_0(Q_\varphi(R^{st}))$. Therefore, the query Q_φ is not $\mathcal{F}$-generic.

We can conclude that Q_φ is $\mathcal{F}$-generic if and only if the sentence $\forall n_1 \cdots \forall n_k$ $\varphi(n_1, \ldots, n_k)$ holds in the natural numbers.

Therefore, if $\mathcal{F}$-genericity would be decidable, also the truth of sentences in the $\forall^*$-fragment of number theory would be decidable. This concludes the proof. □

In the remainder of this section, we show that the first-order queries that are $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic are recursively enumerable, however. We show this by giving sound and complete languages for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic first-order queries, for the groups $(\mathcal{F}_{st}, \mathcal{F}_t)$ mentioned in Section 4.1.

We first define these sound and complete languages, that are point-based logics.

Definition 4.11 (**Spatio-temporal point logic**)**.** Let $\sigma^{st} = \{R_1^{st}, R_2^{st}, \ldots, R_m^{st}\}$ be a spatio-temporal database schema and let Π be a set of predicates. The first-order logic over σ^{st} and Π, denoted by FO(Π, $R_1^{st}, R_2^{st}, \ldots, R_m^{st}$) or FO($\Pi$, σ), can be used as a spatio-temporal query language when variables are interpreted to range over points in $(\mathbb{R}^n \times \mathbb{R})$. The atomic formulas in FO(Π, σ) are equality constraints on point variables, the predicates of Π applied to point variables, and the relation names $R_1^{st}, R_2^{st}, \ldots, R_m^{st}$ from σ^{st} applied to point variables.

A FO(Π, σ^{st})-formula $\varphi(v_1, v_2, \ldots, v_k)$ defines for each spatio-temporal database $\mathfrak{F}$ over σ^{st} a subset $\varphi(\mathfrak{F})$ of $(\mathbb{R}^n \times \mathbb{R})^k$ defined as

$$\{(p_1, \ldots, p_k) \in (\mathbb{R}^n \times \mathbb{R})^k \mid ((\mathbb{R}^n \times \mathbb{R}), \Pi^{(\mathbb{R}^n \times \mathbb{R})}, R_1^{st\mathfrak{F}}, \ldots, R_m^{st\mathfrak{F}}) \models \varphi[p_1, \ldots, p_k]\},$$

where $\varphi[p_1, \ldots, p_k]$ is obtained from the formula $\varphi(v_1, \ldots, v_k)$ by instantiating the variables v_i by the constant points p_i, for $1 \leq i \leq k$.

We now specify what we mean by sound and complete languages. From Definition 3.18, it is clear what it means that a FO(Π, σ)-formula *expresses a constraint database query.*

Definition 4.12 (**Soundness and completeness**)**.** A query language $\mathcal{L}$ is said to be *sound* for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic FO(+, ×, <, 0, 1, σ)-queries on spatio-temporal databases, if formulas in $\mathcal{L}$ only express $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic FO(+, ×, <, 0, 1, σ)-queries on spatio-temporal databases.

A query language $\mathcal{L}$ is said to be *complete* for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic FO(+, ×, <, 0, 1, σ)-queries on spatio-temporal databases, if all $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic FO(+, ×, <, 0, 1, σ)-queries on spatio-temporal databases can be expressed in $\mathcal{L}$.

In the following, we will omit the dependence on the input schema when this is clear from the context, and use the notation FO(Π) for first-order point languages over the predicate set Π.

In the remainder of this section, we first discuss notions of genericity determined by time-independent transformations (Section 4.2.1), afterwards we discuss applications to physics (Section 4.2.2) and we end with genericity for the time-dependent transformations (Section 4.2.3).

4.2.1 Genericity for Time-independent Transformations

In this section, we give a general result concerning $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic queries where $\mathcal{F}_{st}$ is a subgroup of $\mathcal{A}^c_{st}$, the group of time-independent affinities of $(\mathbb{R}^n \times \mathbb{R})$. First, we introduce the point predicates that we will use for the different point languages.

To express the temporal order of events, we use the point predicate **Before**. Let p_1 and p_2 be points in $(\mathbb{R}^n \times \mathbb{R})$. The expression $\mathbf{Before}(p_1, p_2)$ evaluates to true if the time coordinate τ_1 of p_1 is smaller than or equal to the time coordinate τ_2 of p_2. We will often use the derived binary predicate **Cotemp**, which expresses for two points p_1 and p_2 that τ_1 equals τ_2. This predicate can be expressed using the predicate **Before** as follows:

$$\mathbf{Cotemp}(u, v) := \mathbf{Before}(u, v) \wedge \mathbf{Before}(v, u).$$

There are three more other purely temporal predicates: $\mathbf{UnitTime}$, $\mathbf{0_t}$ and $\mathbf{1_t}$. The predicate $\mathbf{UnitTime}(p_1, p_2)$ expresses that the points $p_1, p_2 \in (\mathbb{R}^n \times \mathbb{R})$ have time-coordinates τ_1 and τ_2 such that $|\tau_1 - \tau_2| = 1$. The unary predicates $\mathbf{0_t}$ and $\mathbf{1_t}$ are such that $\mathbf{0_t}(p)$ and $\mathbf{1_t}(p)$ respectively express that the time coordinate τ of the point p equals to zero and to one, respectively.

The following predicates address spatio-temporal relations between points. The point-predicate $\mathbf{Between}^{(n+1)}$ is defined such that $\mathbf{Between}^{(n+1)}(p_1, p_2, p_3)$ expresses that the points p_1, p_2, p_3 in $(\mathbb{R}^n \times \mathbb{R})$ are collinear (in the space $(\mathbb{R}^n \times \mathbb{R})$) and that p_2 is between p_1 and p_3. The predicates $\mathbf{Smaller}_i(p_1, p_2)$ $(1 \leq i \leq n)$ express that the ith spatial coordinate of p_1 is less or equal than the ith spatial coordinate of p_2. The fact $\mathbf{EqDist}^{st}(p_1, p_2, p_3, p_4)$ is true if the distance between the two co-temporal points p_1 and p_2 equals the distance between the two co-temporal points p_3 and p_4. The binary predicate $\mathbf{UnitDist}^{st}$ applied to two points p_1 and p_2 expresses that they are co-temporal and that the (spatial) distance between p_1 and p_2 equals one. Finally, $\mathbf{Pos}^{(n+1)}(p_0, p_1, p_2, \ldots, p_{n+1})$ expresses that the $(n+2)$-tuple $(p_0, p_1, p_2, \ldots, p_{n+1})$ of points in $(\mathbb{R}^n \times \mathbb{R})$ forms a positively oriented $(n+1)$-dimensional coordinate system with p_0 as origin.

Property 4.2.1. The point predicates **Before**, $\mathbf{Between}^{(n+1)}$, $\mathbf{UnitTime}$, $\mathbf{0_t}$, $\mathbf{1_t}$, $\mathbf{Smaller}_i(1 \leq i \leq n)$, $\mathbf{EqDist}^{st}$, $\mathbf{UnitDist}^{st}$ and $\mathbf{Pos}^{(n+1)}$ are all expressible in FO(+, ×, <, 0, 1).

Proof. The FO(+, ×, <, 0, 1)-formulas for the different predicates can be obtained by expressing the constraints on the coordinates of the points satisfying the predicates. We denote the coordinates of a point variable v_i in $(\mathbb{R}^n \times \mathbb{R})$ by $(x_{i_1}, x_{i_2}, \ldots, x_{i_n}, t_i)$, $i = 1, 2, \ldots$. The translation of the expression $\mathbf{Before}(v_1, v_2)$ is $t_1 \leq t_2$ and the translation of $\mathbf{UnitTime}(v_1, v_2)$ equals $\mid t_1 - t_2 \mid = 1$. The translation of $\mathbf{0_t}$ and $\mathbf{1_t}$ is straightforward.

It is well known (e.g. [43, 71]) that the predicates $\mathbf{Between}^{(n+1)}$, $\mathbf{Smaller}_i(1 \leq i \leq n)$, $\mathbf{EqDist}^{st}$ and $\mathbf{UnitDist}^{st}$ are expressible in FO(+, ×, <, 0, 1). For $\mathbf{EqDist}^{st}$ and $\mathbf{UnitDist}^{st}$ it is necessary to use **Before** to express that their arguments should be co-temporal.

The expression $\mathbf{Pos}^{(n+1)}(v_0, v_1, \ldots, v_{n+1})$ is translated into FO(+, ×, <, 0, 1) by expressing that the vectors $v_1 - v_0, \ldots, v_{n+1} - v_0$ are linearly independent and

that the $(n+1) \times (n+1)$-matrix containing their coordinates has a strictly positive determinant. □

$(\mathcal{F}_{st}, \mathcal{F}_t)$	Sets of point predicates $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$
$(\mathcal{A}^c_{st}, \mathcal{A}_t)$	$\{\mathbf{Between}^{(n+1)}, \mathbf{Before}\}$
$(\mathcal{A}^c_{st}, \mathcal{T}_t)$	$\{\mathbf{Between}^{(n+1)}, \mathbf{Before}, \mathbf{UnitTime}\}$
$(\mathcal{A}^c_{st}, \mathrm{Id}_t)$	$\{\mathbf{Between}^{(n+1)}, \mathbf{Before}, \mathbf{UnitTime}, \mathbf{0_t}, \mathbf{1_t}\}$
$(\mathcal{S}^c_{st}, \mathcal{F}_t)$	$\Pi(\mathcal{A}^c_{st}, \mathcal{F}_t) \cup \{\mathbf{EqDist}^{st}\}$
$(\mathcal{I}^c_{st}, \mathcal{F}_t)$	$\Pi(\mathcal{A}^c_{st}, \mathcal{F}_t) \cup \{\mathbf{EqDist}^{st}, \mathbf{UnitDist}^{st}\}$
$(\mathcal{T}^c_{st}, \mathcal{F}_t)$	$\Pi(\mathcal{I}^c_{st}, \mathcal{F}_t) \cup \{\mathbf{Smaller}_i(1 \leq i \leq n), \mathbf{Pos}^{(n+1)}\}$

Table 4.1: An overview of the different sets of point predicates for a number of spatio-temporal genericity notions. In the three last cases $\mathcal{F}_t \in \{\mathcal{A}_t, \mathcal{T}_t, Id_t\}$.

Property 4.2.2 (The elements of $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ are $(\mathcal{F}_{st}, \mathcal{F}_t)$-invariant). Let $(\mathcal{F}_{st}, \mathcal{F}_t)$ be a group and let $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ be a set of point predicates as in Table 4.1. The point predicates in $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ are invariant under elements of $(\mathcal{F}_{st}, \mathcal{F}_t)$.

Proof. First, remark that, if we fix $\mathcal{F}_t$ to be one of $\{\mathcal{A}_t, \mathcal{T}_t, Id_t\}$, then

$$(\mathcal{T}^c_{st}, \mathcal{F}_t) \subset (\mathcal{I}^c_{st}, \mathcal{F}_t) \subset (\mathcal{S}^c_{st}, \mathcal{F}_t) \subset (\mathcal{A}^c_{st}, \mathcal{F}_t).$$

Also, if we fix $\mathcal{F}_{st}$ to be one of $\{\mathcal{A}^c_{st}, \mathcal{S}^c_{st}, \mathcal{I}^c_{st}, \mathcal{T}^c_{st}\}$, then

$$(\mathcal{F}_{st}, Id_t) \subset (\mathcal{F}_{st}, \mathcal{T}_t) \subset (\mathcal{F}_{st}, \mathcal{A}_t).$$

Also, all groups $(\mathcal{F}_{st}, \mathcal{F}_t)$ are subgroups of the affinities of $(\mathbb{R}^n \times \mathbb{R})$. As we already remarked, if a point predicate is invariant for a certain transformation group $(\mathcal{F}_{st}, \mathcal{F}_t)$, it is also invariant for all subgroups of $(\mathcal{F}_{st}, \mathcal{F}_t)$.

We now prove $(\mathcal{F}_{st}, \mathcal{F}_t)$-invariance for each of the predicates in the sets $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ of Table 4.1.

• *The predicate* $\mathbf{Between}^{(n+1)}$ *is invariant under elements of* $(\mathcal{A}^c_{st}, \mathcal{A}_t)$. It is well known that affinities preserve the "betweenness" of points. As all groups listed in Table 4.1 are subgroups of the affinities of $(\mathbb{R}^n \times \mathbb{R})$, the predicate $\mathbf{Between}^{(n+1)}$ is invariant for all those groups.
• *The predicate* **Before** *is invariant under elements of* $(\mathcal{A}^c_{st}, \mathcal{A}_t)$, since the elements of $\mathcal{A}_t$ are monotone bijections of time. As shown in Proposition 4.4, the order on time events is preserved under all strictly monotone increasing bijections of time. The groups $\mathcal{A}_t$, $\mathcal{I}_t$, Id_t are all such bijections.
• *The predicate* **UnitTime** *is invariant under elements of* $(\mathcal{A}^c_{st}, \mathcal{T}_t)$. It is straightforward that all elements of $\mathcal{T}_t$, which are translations in the time direction, preserve the time difference between any two points p_1 and p_2 in $(\mathbb{R}^n \times \mathbb{R})$.
• *The predicates* $\mathbf{0_t}$ *and* $\mathbf{1_t}$ *are invariant under elements of* $(\mathcal{A}^c_{st}, \mathrm{Id}_t)$. It is clear that the identity transformation on the time preserves the fact that a point p in $(\mathbb{R}^n \times \mathbb{R})$ has time coordinate zero or one.

• *The predicate* $\mathbf{EqDist}^{st}$ *is invariant under elements of* $(\mathcal{S}^c_{st}, \mathcal{A}_t)$. It is well known that isometries and scalings (and thus similarities) preserve the fact that the distance between one pair of points equals the distance between a second pair of points. The groups $\mathcal{A}_t$, $\mathcal{T}_t$, Id_t all preserve co-temporality of points.
• *The predicate* $\mathbf{UnitDist}^{st}$ *is invariant under elements of* $(\mathcal{I}^c_{st}, \mathcal{A}_t)$, because isometries are distance preserving transformations.
• *The predicates* $\mathbf{Smaller}_i (1 \leq i \leq n)$ *are invariant under elements of* $(\mathcal{T}^c_{st}, \mathcal{A}_t)$. It is easy to verify that if for two points p_1 and p_2 in $(\mathbb{R}^n \times \mathbb{R})$, $\mathbf{Smaller}_i(p_1, p_2)$ is true for some i in $\{1, \ldots, n\}$, also $\mathbf{Smaller}_i(f(p_1), f(p_2))$ holds for each f in $(\mathcal{T}^c_{st}, \mathcal{F}_t)$, where $\mathcal{F}_t$ is one of $\mathcal{A}_t, \mathcal{T}_t, Id_t$.
• *The predicate* $\mathbf{Pos}^{(n+1)}$ *is invariant under elements of* $(\mathcal{T}^c_{st}, \mathcal{A}_t)$, since translations are orientation-preserving transformations. □

Remark 4.13. From now, all results are valid for underlying dimension $n \geq 2$.

The following theorem follows directly from the proof of Theorem 5.5 [43].

Theorem 4.14 (**Spatial meta-theorem**)**.** Let σ^{st} be a spatio-temporal database schema. Let $\mathcal{F}$ be a subgroup of the affinities of $(\mathbb{R}^n \times \mathbb{R})$. Let Π be a set of point-predicates that contains $\mathbf{Between}^{(n+1)}$. If the predicates in Π are FO(+, ×, <, 0, 1, σ)-expressible and invariant under the transformations of $\mathcal{F}$ and if the fact *"*$(v_0, v_1, \ldots, v_{n+1})$ *is the image of the standard coordinate system of* $(\mathbb{R}^n \times \mathbb{R})$ *under some element* f *of* $\mathcal{F}$*"* is expressible in FO(Π, σ^{st}), then FO(Π, σ^{st}) is sound and complete for the $\mathcal{F}$-generic spatio-temporal database queries that are expressible in FO(+, ×, <, 0, 1, σ).

We now prove the following theorem.

Theorem 4.15 (**Time-independent spatio-temporal meta-theorem**)**.** Let σ^{st} be a spatio-temporal database schema. Let $\mathcal{F}_{st}$ be a subgroup of $\mathcal{A}^c_{st}$ and $\mathcal{F}_t$ a subgroup of $\mathcal{A}_t$. Let $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ be a set of point-predicates that contains $\mathbf{Between}^{(n+1)}$ and **Before**. If the predicates in $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ are FO(+, ×, <, 0, 1, σ)-expressible and invariant under the transformations of $(\mathcal{F}_{st}, \mathcal{F}_t)$ and if the fact *"*$(v_0, v_1, \ldots, v_{n+1})$ *is the image of the standard coordinate system under some element* f *of* $(\mathcal{F}_{st}, \mathcal{F}_t)$*"* is expressible in FO($\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$, σ^{st}), then the logic FO($\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$, σ^{st}) is sound and complete for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic spatio-temporal database queries that are expressible in FO(+, ×, <, 0, 1, σ).

Proof. Let σ^{st} be a spatio-temporal database schema and let σ be the corresponding constraint database schema. We first prove this theorem for $\Pi(\mathcal{A}^c_{st}, \mathcal{A}_t) = \{\mathbf{Between}^{(n+1)}, \mathbf{Before}\}$, using Theorem 4.14. Indeed, it is clear that the group $(\mathcal{A}^c_{st}, \mathcal{A}_t)$ is a subgroup of the affinities of $(\mathbb{R}^n \times \mathbb{R})$. Furthermore, the expression $\mathbf{Before}(u, v)$, is expressible in FO(+, ×, <, 0, 1, σ) (see Property 4.2.1). Also, the predicates $\mathbf{Between}^{(n+1)}$ and **Before** are both invariant under elements of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$ (see Property 4.2.2).

To conclude this part of the proof, we need to show that there is an expression in FO($\{\mathbf{Between}^{(n+1)}, \mathbf{Before}\}$, σ^{st}) that, for $n+2$ arbitrary points $p_0, p_1, \ldots, p_{n+1}$

in $(\mathbb{R}^n \times \mathbb{R})$, states that $(p_0, p_1, \ldots, p_{n+1})$ is the image of the standard coordinate system under some element f of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$. It is known (e.g. [43, 67]) that there exists an expression in the language FO($\{\mathbf{Between}^{(n+1)}\}$, σ^{st}) that, for $n+2$ points p_0, p_1, $, \ldots, p_{n+1}$ of $(\mathbb{R}^n \times \mathbb{R})$, expresses that $(p_0, p_1, \ldots, p_{n+1})$ is the image of the standard $(n+1)$-dimensional coordinate system under some affinity of $(\mathbb{R}^n \times \mathbb{R})$. We refer to this expression as

$$\mathbf{CoSys}_{\mathcal{A}}(v_0, v_1, \ldots, v_{n+1}).$$

Obviously, this formula also belongs to FO($\{\mathbf{Between}^{(n+1)}, \mathbf{Before}\}$, σ^{st}). The expression for the image of the standard coordinate system under some element of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$ is as follows:

$$\mathbf{CoSys}_{(\mathcal{A}^c_{st}, \mathcal{A}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{\mathcal{A}}(v_0, v_1, \ldots, v_{n+1}) \wedge \bigwedge_{i=1}^{n} \mathbf{Cotemp}(v_0, v_i) \wedge \neg \mathbf{Before}(v_{n+1}, v_0).$$

It is easy to verify that any coordinate system that is an image of the standard coordinate system under an element of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$ satisfies this expression. Also, the reverse is true. For clarity, we show this only for $n = 2$ (the general case is analogous).

Any coordinate system (p_0, p_1, p_2, p_3) satisfying the expression $\mathbf{CoSys}_{(\mathcal{A}^c_{st}, \mathcal{A}_t)}(v_0, v_1, v_2, v_3)$ is of the form $p_0 = (a_{0,1}, a_{0,2}, \tau_0)$, $p_1 = (a_{1,1}, a_{1,2}, \tau_0)$, $p_2 = (a_{2,1}, a_{2,2}, \tau_0)$, $p_3 = (a_{3,1}, a_{3,2}, \tau_3)$, where $\tau_0 < \tau_3$ and the determinant

$$\begin{vmatrix} a_{1,1} - a_{0,1} & a_{1,2} - a_{0,2} & 0 \\ a_{2,1} - a_{0,1} & a_{2,2} - a_{0,2} & 0 \\ a_{3,1} - a_{0,1} & a_{3,2} - a_{0,2} & \tau_3 - \tau_0 \end{vmatrix} \neq 0. \qquad (*)$$

Now, we have to show that there exists an element f of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$ such that the image of the standard coordinate system under f equals (p_0, p_1, p_2, p_3). As $(\mathcal{A}^c_{st}, \mathcal{A}_t)$ is a subgroup of the affinities, f is representable by a matrix. It is straightforward to derive that $f = (f_{st}, f_t)$, where

$$f_{st}(x, y, t) = \begin{pmatrix} a_{1,1} - a_{0,1} & a_{2,1} - a_{0,1} \\ a_{1,2} - a_{0,2} & a_{2,2} - a_{0,2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} (a_{3,1} - a_{0,1})t + a_{0,1} \\ (a_{3,2} - a_{0,2})t + a_{0,2} \end{pmatrix}, \text{ and}$$

$$f_t(t) = (\tau_3 - \tau_0)t + \tau_0.$$

It is clear that $(\tau_3 - \tau_0) > 0$ and that, because of the inequality $(*)$, the value of the determinant $\begin{vmatrix} a_{1,1} - a_{0,1} & a_{2,1} - a_{0,1} \\ a_{1,2} - a_{0,2} & a_{2,2} - a_{0,2} \end{vmatrix}$ differs from zero, hence f is an element of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$.

So far, we proved that the language FO($\{\mathbf{Between}^{(n+1)}, \mathbf{Before}\}$, σ^{st}) is sound and complete for the $(\mathcal{A}^c_{st}, \mathcal{A}_t)$-generic queries expressible in FO(+, $\times$, $<$, 0, 1, σ). The fact that any other language FO($\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$, σ), where $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ contains $\mathbf{Between}^{(n+1)}$ and **Before**, is sound and complete for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic FO(+, $\times$, $<$, 0, 1, σ)-queries for each subgroup $(\mathcal{F}_{st}, \mathcal{F}_t)$ of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$, under the conditions stated in Theorem 4.15, follows from Theorem 4.14 together with the first part of this proof. □

Theorem 4.16 (**First-order languages for time-independent transformations**). Let σ^{st} be a spatio-temporal database schema. Let $(\mathcal{F}_{st}, \mathcal{F}_t)$ be a group and let $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ be as in Table 4.1. The point language $\mathrm{FO}(\Pi(\mathcal{F}_{st}, \mathcal{F}_t),\ \sigma^{st})$ is sound and complete for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic queries expressible in $\mathrm{FO}(+,\ \times,\ <,\ 0,\ 1,\ \sigma)$.

Proof. We can apply Theorem 4.15 for all groups in Table 4.1, because they are all subgroups of $(\mathcal{A}^c_{st}, \mathcal{A}_t)$. From Property 4.2.1 and Property 4.2.2, it follows that all predicates are expressible in $\mathrm{FO}(+,\ \times,\ <,\ 0,\ 1)$ and that they are invariant under transformations of the appropriate groups. The only thing left to prove is that, for all groups $(\mathcal{F}_{st}, \mathcal{F}_t)$ from Table 4.1, and for $n+2$ points $v_0, v_1, , \ldots, v_{n+1}$ in $(\mathbb{R}^n \times \mathbb{R})$, the fact "$(v_0, v_1, \ldots, v_{n+1})$ is the image of the standard coordinate system under some element f of $(\mathcal{F}_{st}, \mathcal{F}_t)$" is expressible in $\mathrm{FO}(\Pi(\mathcal{F}_{st}, \mathcal{F}_t),\ \sigma^{st})$. For each group $(\mathcal{F}_{st}, \mathcal{F}_t)$ from Table 4.1, we now give a formula that expresses this fact. The correctness of these formulas is easy to verify.

- *For the group* $(\mathcal{A}^c_{st}, \mathcal{A}_t)$, we already gave a formula in the proof of Theorem 4.15. The desired formula is there denoted by $\mathbf{CoSys}_{(\mathcal{A}^c_{st}, \mathcal{A}_t)}$.
- *For the group* $(\mathcal{A}^c_{st}, \mathcal{T}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{A}^c_{st}, \mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{A}^c_{st}, \mathcal{A}_t)}(v_0, v_1, \ldots, v_{n+1}) \wedge \mathbf{UnitTime}(v_0, v_{n+1}).$$

- *For the group* $(\mathcal{A}^c_{st}, Id_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{A}^c_{st}, Id_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{A}^c_{st}, \mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) \wedge \mathbf{0_t}(v_0) \wedge \mathbf{1_t}(v_{n+1}).$$

Let $\mathcal{F}_t$ be an element of $\{\mathcal{A}_t, \mathcal{T}_t, Id_t\}$.

- *For the groups* $(\mathcal{S}^c_{st}, \mathcal{F}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{S}^c_{st}, \mathcal{F}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{A}^c_{st}, \mathcal{F}_t)}(v_0, v_1, \ldots, v_{n+1}) \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \mathbf{EqDist}^{st}(v_0, v_i, v_0, v_j).$$

- *For the groups* $(\mathcal{I}^c_{st}, \mathcal{F}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{I}^c_{st}, \mathcal{F}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{S}^c_{st}, \mathcal{F}_t)}(v_0, v_1, \ldots, v_{n+1}) \wedge \bigwedge_{i=1}^{n} \mathbf{UnitDist}^{st}(v_0, v_i).$$

- *For the groups* $(\mathcal{T}^c_{st}, \mathcal{F}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{T}^c_{st}, \mathcal{F}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{I}^c_{st}, \mathcal{F}_t)}(v_0, v_1, \ldots, v_{n+1}) \wedge \mathbf{Pos}^{(n+1)}(v_0, v_1, \ldots, v_{n+1}) \wedge \bigwedge_{j=1}^{n} \bigwedge_{i=1}^{n} \mathbf{Smaller}_i(v_0, v_j).$$

□

4.2.2 Applications to Physics

Here, we focus on the transformation groups $(\mathcal{V}_{st}, \mathcal{T}_t)$, $(\mathcal{V}(\mathcal{R})_{st}, \mathcal{T}_t)$, $(\mathcal{AC}_{st}, \mathcal{T}_t)$ and $(\mathcal{AC}(\mathcal{R})_{st}, \mathcal{T}_t)$. To formulate our results we need to define one more point-predicate, namely **EqSpace**. If $p_1 = (a_{1,1}, \ldots, a_{1,n}, \tau_1)$ and $p_2 = (a_{2,1}, \ldots, a_{2,n}, \tau_2)$ are elements of $(\mathbb{R}^n \times \mathbb{R})$, then $\mathbf{EqSpace}(p_1, p_2)$ if and only if $a_{1,i} = a_{2,i}$ for all $1 \leq i \leq n$.

Remark 4.17. The expression

$$\mathbf{EqSpace}(v_1, v_2) := \bigwedge_{i=1}^{n} (\mathbf{Smaller}_i(v_1, v_2) \wedge \mathbf{Smaller}_i(v_2, v_1))$$

is expressible in FO(+, ×, <, 0, 1).

Theorem 4.18 (First-order languages for transformations from physics). Let σ^{st} be a spatio-temporal database schema. Let the groups $(\mathcal{F}_{st}, \mathcal{T}_t)$ and the predicate sets $\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$ be as in Table 4.2. The point language FO($\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$, σ^{st}) is sound and complete for the $(\mathcal{F}_{st}, \mathcal{T}_t)$-generic spatio-temporal queries that are expressible in FO(+, ×, <, 0, 1, σ).

$(\mathcal{F}_{st}, \mathcal{T}_t)$	Set of point predicates $\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$
$(\mathcal{V}_{st}, \mathcal{T}_t)$	{**Between**$^{(n+1)}$, **Before**, **UnitTime**, **EqSpace**}
$(\mathcal{V}(\mathcal{R})_{st}, \mathcal{T}_t)$	{**Between**$^{(n+1)}$, **Before**, **UnitTime**, **EqSpace**, **EqDist**st,**UnitDist**st }
$(\mathcal{AC}_{st}, \mathcal{T}_t)$	{**Between**$^{(n+1)}$, **Before**, **UnitTime**}
$(\mathcal{AC}(\mathcal{R})_{st}, \mathcal{T}_t)$	{**Between**$^{(n+1)}$, **Before**, **UnitTime**, **EqDist**st,**UnitDist**st }

Table 4.2: An overview of the different point-predicate sets for the physical transformation groups.

Proof. The transformation groups $(\mathcal{F}_{st}, \mathcal{T}_t)$ of Table 4.2 are all subgroups of the group $(\mathcal{A}^c_{st}, \mathcal{A}_t)$. Furthermore, the predicates of $\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$ are expressible in FO(+, ×, <, 0, 1) (see Property 4.2.1 and Remark 4.17). Straightforward geometrical and physical arguments show that all predicates are invariant under the appropriate transformation groups. We can now apply Theorem 4.15. We only have to verify that it is possible to express in the languages FO($\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$, σ^{st}) that a coordinate system is the image of the standard $(n+1)$-dimensional coordinate system under an element of $(\mathcal{F}_{st}, \mathcal{T}_t)$. We now give, for each group $(\mathcal{F}_{st}, \mathcal{F}_t)$ from Table 4.2, the expression for the fact that $(v_0, v_1, \ldots, v_{n+1})$ is the image of the standard coordinate system under some element f of $(\mathcal{F}_{st}, \mathcal{F}_t)$.

The correctness of these expressions is easy to verify.

- *For the group* $(\mathcal{V}_{st}, \mathcal{T}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{V}_{st}, \mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{A}_{st}, \mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) \wedge \mathbf{EqSpace}(v_0, v_{n+1}),$$

because elements of this group map the origin $(0,\ldots,0,0)$ and the unit vector in the time-direction $(0,\ldots,0,1)$ of the standard coordinate system of $(\mathbb{R}^n \times \mathbb{R})$ onto points which have equal spatial coordinates.

• *For the group* $(\mathcal{V}(\mathcal{R})_{st}, \mathcal{T}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{V}(\mathcal{R})_{st},\mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{I}_{st},\mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) \wedge \mathbf{EqSpace}(v_0, v_{n+1}).$$

• *For the group* $(\mathcal{AC}_{st}, \mathcal{T}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{AC}_{st},\mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{A}_{st},\mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}).$$

• *For the group* $(\mathcal{AC}(\mathcal{R})_{st}, \mathcal{T}_t)$, we have

$$\mathbf{CoSys}_{(\mathcal{AC}(\mathcal{R})_{st},\mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}) := \mathbf{CoSys}_{(\mathcal{I}_{st},\mathcal{T}_t)}(v_0, v_1, \ldots, v_{n+1}).$$

□

Next, we illustrate the languages summarized in Table 4.1 and Table 4.2 on the appropriate examples of Section 4.1.5.

Example 4.19. We give the FO($\{\mathbf{Between}^{(n+1)}, \mathbf{Before}, \mathbf{UnitTime}, \mathbf{EqSpace}\}$, $\sigma^{st} = \{carA, carB\}$)-query Q_2' equivalent to the $(\mathcal{V}_{st}, \mathcal{T}_t)$-generic query of Example 4.6: *Give the places and time moments where carA is standing still at that spot for at least* 300 *more seconds.*

Remember that we assumed before that time is measured in seconds and distance is measured in meters. We first remark that the fact that one point is a constant number of seconds before another, can be expressed using **UnitTime** and **Before**. We illustrate this for an easy example where one point is 3 seconds after another:

$$3sec(u,v) := \exists w_1\, \exists w_2\, (\mathbf{Before}(u, w_1, \wedge)\mathbf{Before}(w_1, w_2, \wedge)\mathbf{Before}(w_2, v, \wedge) \mathbf{UnitTime}(u, w_1, \wedge)\mathbf{UnitTime}(w_1, w_2, \wedge)\mathbf{UnitTime}(w_2, v,)).$$

Now we give the expression for Q_2':

$$carA(u) \wedge \exists v\, (300sec(u,v) \wedge \forall w\, ((\mathbf{Before}(u, w, \wedge)\mathbf{Before}(w, v, \wedge)carA(w)) \rightarrow \mathbf{EqSpace}(u, w))).$$

□

Example 4.20. We give the FO($\{\mathbf{Between}^{(n+1)}, \mathbf{Before}, \mathbf{UnitTime}\}$, $\sigma^{st} = \{carA, carB\}$)-query Q_5' equivalent to the $(\mathcal{AC}_{st}, \mathcal{T}_t)$-generic query of Example 4.9: *Did car A encounter any empty roads? I.e., were there parts of its trajectory where it could drive at constant speed for at least 6000 seconds.*

$$\exists u\, \exists v\, (carA(u) \wedge carA(v) \wedge 6000sec(u,v) \wedge \forall w\, ((carA(w) \wedge \mathbf{Before}(u, w, \wedge)\mathbf{Before}(w, v,)) \rightarrow (\mathbf{Between}^{(n+1)}(u, v, w)))).$$

□

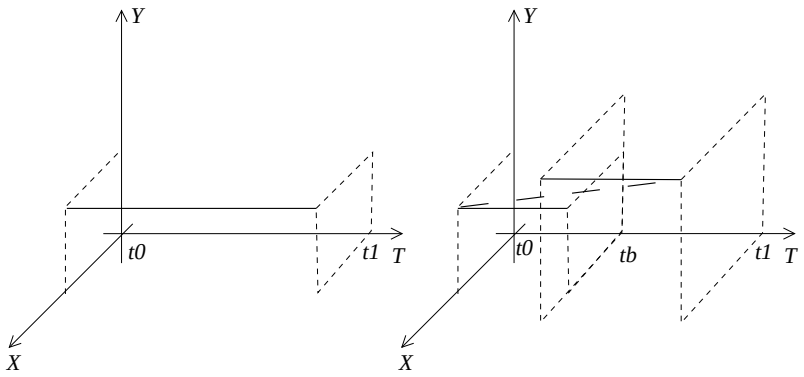


Figure 4.1: The elements of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ do not preserve betweenness of points.

4.2.3 Genericity for Time-dependent Transformations

Here, we focus on notions of genericity determined by time-dependent transformations. Our first result in this context shows that we can restrict our attention, without loss of generality, to piece-wise constant transformations.

Proposition 4.21. Let Q be a spatio-temporal query expressible in FO(+, ×, <, 0, 1) and let the group $\mathcal{F}_{st}$ be $\mathcal{A}_{st}$, $\mathcal{S}_{st}$, $\mathcal{I}_{st}$ or $\mathcal{T}_{st}$ and the group $\mathcal{F}_t$ be $\mathcal{A}_t$, $\mathcal{T}_t$ or Id_t. Then Q is $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic if and only if it is $(\mathcal{F}^f_{st}, \mathcal{F}_t)$-generic.

Although we postpone the proof of this proposition until the end of this section, it allows us to focus on subgroups of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$.

We first look at the group $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ and next on its subgroups. It will become clear later, that the proof strategy for these groups is analogous to that for the group $(\mathcal{A}^f_{st}, \mathcal{A}_t)$.

It is important to note that for $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ and its subgroups, we cannot apply Theorem 4.15. Indeed, it heavily relies on the fact that, using the predicate $\mathbf{Between}^{(n+1)}$, it can be expressed that $n+2$ points form an affine coordinate system for the space $(\mathbb{R}^n \times \mathbb{R})$, and also that some points represent the coordinates of another point, relative to such an affine coordinate system (the latter is a straightforward consequence of the former). When using the transformation group $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ or one of its subgroups, the predicate $\mathbf{Between}^{(n+1)}$ is too strong. Indeed, transformations of the group $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ do not preserve "betweenness" in $(n+1)$-dimensional space of points with different time coordinates. Therefore, the notion of collinearity in $(n+1)$-dimensional space can no longer be used. Figure 4.1 illustrates this with a line (left) and the image of the line under some transformation $\alpha = (\alpha_{st}, \alpha_t)$ in $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ for which α_t is the identity function and α_{st} equals the identity in the time interval $[t_0, t_b[$ and is a constant translation of space for the interval $[t_b, t_1]$. In the left part of Figure 4.1, it is true that all points different from the endpoints at time moments t_0 and t_1 lie between the endpoints. For the right part of Figure 4.1 this is not true (the dashed line connecting the end points indicates all points between them.)

However, as we want our language to be able to express all first-order $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-generic queries, somehow there needs to be a link between an $(n+1)$-dimensional point and its coordinates. It will become more clear later that, although we cannot express projection along the time axis, this link can be expressed using the predicates $\mathbf{Between}^{\mathrm{Cotemp}}$, **Before** and a new predicate, $\mathbf{EqCr}^{ST}$. The predicate **Before** has already been introduced in Section 4.2.1. The expression $\mathbf{Between}^{\mathrm{Cotemp}}(p,q,r)$ states, for three points $p, q, r \in (\mathbb{R}^n \times \mathbb{R})$, that they are co-temporal, collinear in the space $\mathbb{R}^n$ and that q is between p and r. We also introduce a new 6-ary predicate, $\mathbf{EqCr}^{ST}$. For six points $p_1, p_2, p_3, q_1, q_2, q_3 \in (\mathbb{R}^n \times \mathbb{R})$, $\mathbf{EqCr}^{ST}(p_1, p_2, p_3, q_1, q_2, q_3)$ expresses that the cross ratio of the three co-temporal and collinear points p_1, p_2 and p_3 equals the cross ratio of the time coordinates τ_{q_1}, τ_{q_2} and τ_{q_3} of the points q_1, q_2 and q_3. The cross ratio of three collinear points p, q, r is $\frac{|pq|}{|pr|}$. It is well known that the cross ratio is invariant under affine transformations.

For example, in $(\mathbb{R}^n \times \mathbb{R})$,

$$\mathbf{EqCr}^{ST}((0,0,0),(1,1,0),(2,2,0),(0,0,0),(0,0,1),(0,0,2))$$

holds, since the former three points have a cross ratio of $\frac{\sqrt{2}}{2\sqrt{2}}$ and the latter three points have a cross ratio of $\frac{1}{2}$.

For ease of use, we will often use the predicates $\mathbf{EqCR}^{\mathbf{s}}$ for the cross-ratio of spatial coordinates, and $\mathbf{EqCR}^{\mathbf{t}}$ for the cross-ratio of temporal coordinates. Both predicates can be expressed using $\mathbf{EqCr}^{ST}$:

$$\begin{aligned}\mathbf{EqCR}^{\mathbf{s}}(u_1,u_2,u_3,v_1,v_2,v_3) := \exists w_1\, \exists w_2\, \exists w_3 \\ (\mathbf{EqCr}^{ST}(u_1,u_2,u_3,w_1,w_2,w_3) \wedge \mathbf{EqCr}^{ST}(v_1,v_2,v_3,w_1,w_2,w_3)),\end{aligned}$$

and

$$\begin{aligned}\mathbf{EqCR}^{\mathbf{t}}(u_1,u_2,u_3,v_1,v_2,v_3) := \exists w_1\, \exists w_2\, \exists w_3 \\ (\mathbf{EqCr}^{ST}(w_1,w_2,w_3,u_1,u_2,u_3) \wedge \mathbf{EqCr}^{ST}(w_1,w_2,w_3,v_1,v_2,v_3)).\end{aligned}$$

Next, we present the main theorem of this section. The proof is composed of three lemmas, as explained below.

Theorem 4.22 (FO($\{\mathbf{Between}^{\mathbf{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$, σ^{st}) **is sound and complete for the** $(\mathcal{A}^f_{st}, \mathcal{A}_t)$**-generic** FO($+$, $\times$, $<$, 0, 1, σ) **queries**)**.** Let σ^{st} be a spatio-temporal database schema. The language FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$, σ^{st}) is sound and complete for the $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-generic spatio-temporal queries that are expressible in FO($+$, $\times$, $<$, 0, 1, σ).

For the remainder of this section, we will abbreviate the set $\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$ by $\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$.

We prove Theorem 4.22 by three lemmas. First, the soundness is addressed in Lemma 4.23. Next, we prove completeness in two steps: Lemma 4.24 shows that every FO($+$, $\times$, $<$, 0, 1, σ)-formula can be converted into a FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-formula, parameterized by a set of coordinate systems and Lemma 4.25 shows then

that every $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-generic spatio-temporal query that is expressible in FO(+, ×, <, 0, 1, σ) can be converted into an equivalent query expressible in the language FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st}).

Lemma 4.23 (**Soundness of** FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})). Let σ^{st} be a spatio-temporal database schema. The language FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st}) is sound for the $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-generic spatio-temporal queries expressible in FO(+, ×, <, 0, 1, σ).

Proof. Soundness is proved in two steps. First, we show that every FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-formula is equivalently expressible in FO(+, ×, <, 0, 1, σ) and afterwards that every FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-formula is invariant under elements of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$. Both are proved by induction on the structure of FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-formulas.

Every FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})*-formula is expressible in* FO(+, ×, <, 0, 1, σ).

The atomic formulas of FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st}) are equality on point variables, the predicates $\mathbf{Between}^{\mathrm{Cotemp}}$, **Before**, $\mathbf{EqCr}^{ST}$ and formulas of the type $R^{st}(v_1, \ldots, v_\ell)$, where R^{st} is a relation name from σ^{st}, with arity ℓ. We now describe, for each of the above types of atomic formulas, how they can be translated into FO(+, ×, <, 0, 1, σ). A point variable v occurring in a FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-formula is translated into real variables $x^v_1, \ldots, x^v_n, t^v$. Equality between two point variables is then expressed in FO(+, ×, <, 0, 1, σ) by requiring that all corresponding coordinates of the two point variables are equal.

We already know that the predicate **Before** is expressible in FO(+, ×, <, 0, 1). The predicate $\mathbf{Between}^{\mathrm{Cotemp}}$ is translated in a similar way as $\mathbf{Between}^{(n+1)}$, with the additional restriction that the time coordinates of the variables should be the same.

The formula $\mathbf{EqCr}^{ST}(u_1, u_2, u_3, v_1, v_2, v_3)$ is translated as the conjunction of (i) the translation of the expression $\mathbf{Collinear}^{\mathbf{Cotemp}}(u_1, u_2, u_3)$, which is equal to

$$\mathbf{Between}^{\mathrm{Cotemp}}(u_1, u_2, u_3) \vee \mathbf{Between}^{\mathrm{Cotemp}}(u_2, u_1, u_3) \vee \mathbf{Between}^{\mathrm{Cotemp}}(u_1, u_3, u_2)$$

and (ii) the formula

$$(t^{v_3} - t^{v_1})^2 \sum_{i=1}^{n} (x_i^{u_1} - x_i^{u_2})^2 = (t^{v_2} - t^{v_1})^2 \sum_{i=1}^{n} (x_i^{u_1} - x_i^{u_3})^2.$$

We translate formulas of the type $R^{st}(v_1, \ldots, v_\ell)$, where R^{st} is a relation name from σ^{st} with arity ℓ, by the formula $R(x_1^{v_1}, \ldots, x_n^{v_1}, t^{v_1}, \ldots, x_1^{v_\ell}, \ldots, x_n^{v_\ell}, t^{v_\ell})$. Compositions of atomic formulas by logical connectives and quantifiers are translated in a natural way.

Every FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})*-formula is invariant for elements of the group* $(\mathcal{A}^f_{st}, \mathcal{A}_t)$.

The only non-trivial part here is showing that all point predicates are $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-invariant. The predicate **Before** is invariant for all transformations $f = (f_1, f_2, \ldots, f_n, f_t)$, that map $(\mathbb{R}^n \times \mathbb{R})$ to $(\mathbb{R}^n \times \mathbb{R})$, such that f_t is a strictly monotone increasing

bijection of t alone (Proposition 4.4). Since all elements of $\mathcal{A}_t$ are such bijections, this condition is satisfied for $(\mathcal{A}^f_{st}, \mathcal{A}_t)$. It is well known that affinities preserve the cross-ratio of three points. Because the predicate $\mathbf{Between}^{\mathrm{Cotemp}}$ requires its parameters to be co-temporal (which is preserved by elements of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$), these co-temporal points will be transformed by the same affinity and hence their cross-ratio is preserved. Also the predicate $\mathbf{EqCr}^{\check{S}T}$ is invariant under elements of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$, because the group $\mathcal{A}^f_{st}$ preserves the cross-ratio between the spatial coordinates of co-temporal points and the group $\mathcal{A}_t$ preserves the cross-ratio between time coordinates. □

We now show that every FO(+, ×, <, 0, 1, σ)-formula can be converted into a FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-formula, which is parameterized by a finite set of coordinate systems.

A coordinate system in a n-dimensional hyperplane of $(\mathbb{R}^n \times \mathbb{R})$, orthogonal to the time axis will be referred to as a *spatial coordinate system* and a coordinate system on the time-axis will be referred to as a *temporal coordinate system.*

If p, q and r are collinear points in $(\mathbb{R}^n \times \mathbb{R})$, then we denote by $\frac{\vec{pq}}{\vec{pr}}$ the real number α such that $\vec{pq} = \alpha\vec{pr}$.

Lemma 4.24 (Translation of FO(+, ×, <, 0, 1, σ) into FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})). Let σ^{st} be a spatio-temporal database schema and let the underlying dimension be n. For every FO(+, ×, <, 0, 1, σ)-formula

$$\psi(x_1, x_2, \ldots, x_m, t_1, \ldots, t_\ell),$$

there exists a FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-formula

$$\psi^{st}(u_{t_O}, u_{t_E}, u_{0,0}, u_{0,1}, \ldots, u_{0,n}, \ldots, u_{\ell,0}, u_{\ell,1}, \ldots, u_{\ell,n}, v_1, v_2, \ldots, v_k),$$

where ℓ is the number of variables occurring in the formula that refer to a time dimension and where k is the total number of free variables of ψ, *i.e.*, $k = m + \ell$.

Furthermore, for each spatio-temporal database $\mathfrak{F}$ over σ^{st}, for each set of spatial coordinate systems $(p_{i,0}, p_{i,1}, \ldots, p_{i,n})$, $i = 0, \ldots, \ell$ of the spatial component of $(\mathbb{R}^n \times \mathbb{R})$, for each temporal coordinate system (p_{t_O}, p_{t_E}) of the temporal component of $(\mathbb{R}^n \times \mathbb{R})$, and for all points $q_1, q_2, \ldots, q_k$ on the line $p_{0,0}p_{0,1}$:

$$((\mathbb{R}^n \times \mathbb{R}), \Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)^{(\mathbb{R}^n \times \mathbb{R})}, \mathfrak{F}) \models \psi^{st}[p_{t_O}, p_{t_E}, p_{0,0}, p_{0,1}, \ldots, p_{0,n}, \ldots, p_{\ell,0}, p_{\ell,1}, \ldots, p_{\ell,n}, q_1, q_2, \ldots, q_k]$$

if and only if

$$(\mathbb{R}, +, \times, <, 0, 1, \alpha(canDB(\mathfrak{F})) \models \psi[\frac{\overrightarrow{p_{0,0}q_1}}{\overrightarrow{p_{0,0}p_{0,1}}}, \frac{\overrightarrow{p_{0,0}q_2}}{\overrightarrow{p_{0,0}p_{0,1}}}, \ldots, \frac{\overrightarrow{p_{0,0}q_k}}{\overrightarrow{p_{0,0}p_{0,1}}}],$$

where $\alpha = (\alpha_{st}, \alpha_t)$ is an element of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ such that $(p_{0,0}, \ldots, p_{0,n})$ is mapped by α_{st} onto the standard spatial coordinate system in the hyperplane $\mathbb{R}^n \times \{(0, \ldots, 0, 0)\}$ of $(\mathbb{R}^n \times \mathbb{R})$, and each spatial coordinate system $(p_{i,0}, p_{i,1}, \ldots, p_{i,n})(i = 1 \ldots, \ell)$ is mapped on the standard coordinate system in the hyperplane at time $\mathbb{R}^n \times \{\alpha(\tau_{p_{i,0}})\}$ where the temporal part α_t of α is the unique time-affinity which maps τ_{p_O} to 0 and τ_{p_E} to 1.

Proof. Let ψ be a FO(+, $\times$, <, 0, 1, σ)-formula. We assume that ψ is in prenex normal form. We now describe the translation of ψ into a formula ψ^{st} of FO(Π($\mathcal{A}^f_{st}, \mathcal{A}_t$), σ^{st}) (by induction on its structure). In this translation, first the quantifier-free part of ψ is translated and the quantifiers are later added in the obvious way.

To start with, a 2-dimensional "computation plane" is chosen that is used to simulate real variables and constants and all the polynomial equations, polynomial equalities and inequalities.

• *The choice of a computation plane.* First of all, two moments in time u_{t_O} and u_{t_E} (time moments are simulated in ψ^{st} by variables in $(\mathbb{R}^n \times \mathbb{R})$) are chosen such that $\neg$**Before**(u_{t_E}, u_{t_O}). They form a temporal coordinate system; the formula describing this is as follows:

$$\mathbf{TCoSys}_{\mathcal{A}_t}(u_1, u_2) := \neg\mathbf{Before}(u_2, u_1).$$

Next, in the hyperplane of points co-temporal with u_{t_O}, $n+1$ points $u_{0,0}, u_{0,1}, \ldots,$ $u_{0,n}$ are chosen such that they form an affine coordinate system for the hyperplane co-temporal with u_{t_O}. The predicate $\mathbf{CoSys}^n_{\mathcal{A}}$, expressing this, is similar the the previously introduced predicate $\mathbf{CoSys}_{\mathcal{A}}$ (see the proof of Theorem 4.16), except that some constraints are added that express that the points should be co-temporal.

As the variables u_{t_O}, u_{t_E}, $u_{0,0}, u_{0,1}, \ldots, u_{0,n}$ represent arbitrary points (up to the mentioned restrictions), they parameterise the translation of ψ. To start with, ψ^{st} will contain the sub-formula ψ^{st}_{comp}, defined as

$$\begin{aligned}\psi^{st}_{\text{comp}}(u_{t_O}, u_{t_E}, u_{0,0}, u_{0,1}, \ldots, u_{0,n}) := \mathbf{TCoSys}_{\mathcal{A}}(u_{t_O}, u_{t_E}) \\ \wedge \mathbf{CoSys}^n_{\mathcal{A}}(u_{0,0}, u_{0,1}, \ldots, u_{0,n}) \wedge \mathbf{Cotemp}(u_{t_O}, u_{0,0}),\end{aligned}$$

as a conjunct.

We will use the 2-dimensional plane through the points $u_{0,0}, u_{0,1}$ and $u_{0,2}$ as a "computation plane". The idea is that we will simulate real variables and constants by points on the line through $u_{0,0}$ and $u_{0,1}$ and that addition and multiplication of real terms are simulated by FO(Π) expressions in the plane through $u_{0,0}, u_{0,1}$ and $u_{0,2}$.

• *The translation of terms and atomic formulas.* A quantifier-free FO(+, $\times$, <, 0, 1, σ)-formula may contain the following terms and atomic sub-formulas: real variables; the constants 0 and 1; polynomial constraints; and relation predicates where the relation names from σ are used. We translate each separately.

− *The translation of real variables.* Each real variable x appearing in the formula ψ is translated into a spatio-temporal variable v. Also, ψ^{st} will contain a conjunct

$$\psi^{st}_{\text{var}}(v) := \mathbf{Collinear}^n(u_{0,0}, u_{0,1}, v),$$

expressing that v is in the computation plane on the line connecting $u_{0,0}$ and $u_{0,1}$. The idea is that a real variable x taking concrete value a, is simulated by requiring that v is such that $\frac{\overrightarrow{u_{0,0}v}}{\overrightarrow{u_{0,0}u_{0,1}}}$ equals a.

− *The translation of the constants* 0 *and* 1. The real constants 0 and 1 that may appear in ψ are translated into $u_{0,0}$ and $u_{0,1}$ respectively.

– *The translation of polynomial constraints.* The arithmetic operations (addition and multiplication) on real terms will be simulated in the computation plane ($u_{0,0}, u_{0,1}$, $u_{0,2}$) It was shown by Tarski [67] (the results of Tarski were also used in [43]) that all arithmetic operations on points that are located on the line through $u_{0,0}$ and $u_{0,1}$ can be simulated in the plane $(u_{0,0}, u_{0,1}, u_{0,2})$ using only the construct $\mathbf{Between}^{(n+1)}$ (and therefore also using $\mathbf{Between}^{\mathrm{Cotemp}}$). Hence, a sub-formula $p(x_1, \ldots, x_m) > 0$, with p a polynomial with integer coefficients, using the translation of the real variables $x_1, \ldots, x_m$ in point variables $v_1, \ldots, v_m$, is translated into $\psi^{st}_{\mathrm{poly}}(u_{0,0}, u_{0,1}, u_{0,2}, v_1, \ldots,$ $v_m)$, defined using the predicate $\mathbf{Between}^{\mathrm{Cotemp}}$.

The correctness of the three above translations can be demonstrated as that of the similar translations in [43].

– *The translation of relation predicates.* A sub-formula of ψ of type $R(x_{1,1}, \ldots, x_{1,n}$, $x_{1,t}, \ldots, x_{m,1}, \ldots, x_{m,n}, x_{m,t})$, where $R \in \sigma$ and where m is the arity of R^{st} in σ^{st}, is translated into a formula

$$R^{st}(v_1, \ldots, v_m)$$

and ψ^{st} has a conjunct expressing that the point variables $v_{1,1}, \ldots, v_{1,n}, v_{1,t}, \ldots, v_{m,1}$, $\ldots, v_{m,n}, v_{m,t}$, that are the translations of $x_{1,1}, \ldots, x_{1,n}, x_{1,t}, \ldots, x_{m,1}, \ldots, x_{m,n}, x_{m,t}$, are the coordinates of $v_1, \ldots, v_m$ respectively. For the moment, we assume that the variables $x_{i,t}$ and $x_{j,t}$ are different for $1 \leq i < j \leq m$ and later show how to deal with the general case. Indeed, recall that each variable $x_{i,j}$ $(1 \leq i \leq m, 1 \leq j \leq n)$ and $x_{i,t}$ $(1 \leq i \leq m)$ is already translated into a point variable $v_{i,j}$ and $v_{i,t}$, which are all collinear with $u_{0,0}$ and $u_{0,1}$. To express the link between the coordinates of point variables $v_1, \ldots, v_m$ and the point variables $v_{i,j}$ and $v_{i,t}$, we proceed as follows. We associate with each point variable v_i $(1 \leq i \leq m)$ the following set of point variables:

1. $(n+1)$ point variables $u_{i,0}, \ldots, u_{i,n}$ representing an n-dimensional coordinate system which is co-temporal with v_i; and

2. n point variables $v'_{i,j}$ which are collinear with $u_{i,0}$ and $u_{i,1}$, such that $v'_{i,j}$ represents the jth coordinate of v_i with respect to the coordinate systems specified by $u_{i,0}, \ldots, u_{i,n}$, and such that the coordinate of $v'_{i,j}$, on the line through $u_{i,0}$ and $u_{i,1}$, gives the same cross ratio with respect to these points as the coordinate of $v_{i,j}$, on the line through $u_{0,0}$ and $u_{0,1}$, gives with respect to these points, *i.e.*, $\frac{\overrightarrow{u_{i,0}v'_{i,j}}}{\overrightarrow{u_{i,0}u_{i,1}}} = \frac{\overrightarrow{u_{i,0}v_{i,j}}}{\overrightarrow{u_{0,0}u_{0,1}}}$.

As explained before, the first set of $(n+1)$ point variables can be defined using the expression

$$\mathbf{CoSys}^n_{\mathcal{A}}(u_{i,0}, u_{i,1}, \ldots, u_{i,n}) \wedge \mathbf{Cotemp}(u_{i,0}, v_i).$$

For the second set of n point variables, we first observe that from [43], we know that we can express, using $\mathbf{Between}^{\mathrm{Cotemp}}$, that n point variables $v'_{i,1}, \ldots, v'_{i,n}$ represent the spatial coordinates of the point variable v_i relative to a chosen spatial coordinate system (in this case, the coordinate system specified by $u_{i,0}, \ldots, u_{i,n}$). In order to establish the link between the point variables $v'_{i,j}$ in the plane specified by $u_{i,0}, \ldots, u_{i,n}$ and the point variables $v_{i,j}$ in the computation plane we need to use the

predicate $\mathbf{EqCR^s}$. The predicate $\mathbf{EqCR^s}$ performs a transformation between the affine coordinate systems at two different time moments, and so connects each $v'_{i,j}$ to a $v_{i,j}$ $(i = 1, \ldots, m, j = 1, \ldots, n)$. Remark that all $v'_{i,j}$ are collinear with $u_{i,0}$ and $u_{i,1}$, and that all $v_{i,j}$ are collinear with $u_{0,0}$ and $u_{0,1}$. Therefore, $\mathbf{EqCR^s}$ can be used to express this equality of cross ratios.

Until now, we only considered the spatial coordinates. To link the temporal variables $v_{i,t}$ to the temporal coordinate of v_i, we use the expression $\mathbf{EqCr}^{ST}(u_{0,0}, u_{0,1}, v_{i,t}, u_{t_O}, u_{t_E}, v_i)$. Recall that the predicate $\mathbf{EqCr}^{ST}$ can be used to relate the cross ratio of points on the time axis to the cross ratio of points, representing coordinates on the line through $u_{0,0}$ and $u_{0,1}$, and thus connects each v_i to a $v_{i,t}$ $(i = 1, \ldots, m)$.

Putting everything together results in the expression ψ^{st}_{rel}:

$$\begin{aligned}
\exists v_1\, \exists v_2\, \ldots\, \exists v_m\, (R^{st}(v_1, v_2, \ldots, v_m) \wedge \bigwedge_{i=1}^{m} \mathbf{CoSys}^n_{\mathcal{A}}(u_{i,0}, u_{i,1}, \ldots, u_{i,n}) \\
\wedge \bigwedge_{i=1}^{m} \mathbf{Cotemp}(u_{i,0}, v_i) \wedge \exists v'_{1,1}\, \ldots\, \exists v'_{1,n}\, \ldots\, \exists v'_{m,1}\, \ldots\, \exists v'_{m,n} \\
(\bigwedge_{i=1}^{m} \mathbf{Coordinates}^n(u_{i,0}, u_{i,1}, \ldots, u_{i,n}, v'_{i,1}, \ldots, v'_{i,n}, v_i) \\
\wedge \bigwedge_{i=1}^{m} \bigwedge_{j=1}^{n} \mathbf{EqCR^s}(u_{0,0}, u_{0,1}, v_{i,j}, u_{i,0}, u_{i,1}, v'_{i,j}) \\
\wedge \bigwedge_{i=1}^{m} \mathbf{EqCr}^{ST}(u_{0,0}, u_{0,1}, v_{i,t}, u_{t_O}, u_{t_E}, v_i)))
\end{aligned}$$

where $\mathbf{Coordinates}^n(u_{i,0}, \ldots, u_{i,n}, v'_{i,1}, \ldots, v'_{i,n}, v_i)$ expresses for each $(1 \leq j \leq n)$ that $v'_{i,j}$ is represents the jth coordinate of v_i with respect to the coordinate systems specified by $u_{i,0}, \ldots, u_{i,n}$.

We now show the correctness of the above translation of a relation predicate. We have to prove that for each spatio-temporal database $\mathfrak{F}$, and for any points p_{t_O}, p_{t_E}, $p_{0,0}, \ldots, p_{0,n}, \ldots, p_{m,0}, \ldots, p_{m,n}, q_{1,1}, \ldots, q_{1,n}, q_{1,t}, \ldots, q_{m,1}, \ldots, q_{m,n}, q_{m,t}$:

$$\begin{aligned}
((\mathbb{R}^n \times \mathbb{R}), \Pi^{(\mathbb{R}^n \times \mathbb{R})}, \mathfrak{F}) \models \psi^{st}_{\text{rel}}[p_{t_O}, p_{t_E}, p_{0,0}, \ldots, p_{0,n}, \ldots, p_{m,0}, \ldots, p_{m,n}, \\
q_{1,1}, \ldots, q_{1,n}, q_{1,t}, \ldots, q_{m,1}, \ldots, q_{m,n}, q_{m,t}]
\end{aligned}$$

if and only if

$$\begin{aligned}
(\mathbb{R}, +, \times, 0, 1, \alpha(\overline{\mathfrak{F}})) \models R[\frac{\overrightarrow{p_{0,0}q_{1,1}}}{\overrightarrow{p_{0,0}p_{0,1}}}, \ldots, \frac{\overrightarrow{p_{0,0}q_{1,n}}}{\overrightarrow{p_{0,0}p_{0,1}}}, \frac{\overrightarrow{p_{0,0}q_{1,t}}}{\overrightarrow{p_{0,0}p_{0,1}}}, \ldots, \\
\frac{\overrightarrow{p_{0,0}q_{m,1}}}{\overrightarrow{p_{0,0}p_{0,1}}}, \ldots, \frac{\overrightarrow{p_{0,0}q_{m,n}}}{\overrightarrow{p_{0,0}p_{0,1}}}, \frac{\overrightarrow{p_{0,0}q_{m,t}}}{\overrightarrow{p_{0,0}p_{0,1}}}],
\end{aligned}$$

where $\alpha = (\alpha_{st}, \alpha_t) \in (\mathcal{A}^f_{st}, \mathcal{A}_t)$ is the affinity which maps $(p_{0,0}, \ldots, p_{0,n})$ to the spatial standard basis at time $\tau_0 = 0$, $(p_{i,0}, \ldots, p_{i,n})$ to the spatial standard basis at time

$\tau_i = \alpha(\tau_{p_{i,0}})$, where α_t is uniquely determined on the time axis by $\alpha_t(\tau_{p_O}) = 0$ and $\alpha_t(p_{t_E}) = 1$. Note that by assumption, $x_{i,t} \neq x_{j,t}$ for $(1 \leq i < j < m)$ and hence also $\tau_{p_{i,0}}$ and $\tau_{p_{j,0}}$, and consequently $\tau_i \neq \tau_j$ for $(1 \leq i < j < m)$. This condition is essential to ensure that α_t exists and is well defined. Indeed, suppose that there exists an i and j such that $\tau_{p_{i,0}} = \tau_{p_{j,0}}$ and hence $\tau_i = \tau_j$. Then we would require that α maps two possibly different co-temporal coordinate systems $(p_{i,0}, \ldots, p_{i,n})$ and $(p_{j,0}, \ldots, p_{j,n})$ the same standard basis. This can clearly not be done by a $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-generic query.

We know that the formula ψ^{st}_{rel} is true for the points $p_{t_O}, p_{t_E}, p_{0,0}, \ldots, p_{0,n}, \ldots, p_{m,0}, \ldots, p_{m,n}, q_{1,1}, \ldots, q_{1,n}, q_{1,t}, \ldots, q_{m,1}, \ldots, q_{m,n}, q_{m,t}$ if and only if points $p_1, \ldots, p_m, q'_{1,1}, \ldots, q'_{1,n}, \ldots, q'_{m,1}, \ldots, q'_{m,n}$ exist such that for each $i = 1, \ldots, m$:

$$\overrightarrow{p_{0,0}p_i} = \overrightarrow{p_{i,0}p_i} + \overrightarrow{p_{0,0}p_{i,0}} = \sum_{j=1}^{n} \frac{\overrightarrow{p_{i,0}q'_{i,j}}}{\overrightarrow{p_{i,0}p_{i,j}}} \overrightarrow{p_{i,0}p_{i,j}} + \overrightarrow{p_{0,0}p_{i,0}}, \tag{4.1}$$

and the following equations hold:

$$\frac{\overrightarrow{p_{i,0}q'_{i,j}}}{\overrightarrow{p_{i,0}p_{i,1}}} = \frac{\overrightarrow{p_{0,0}q_{i,j}}}{\overrightarrow{p_{0,0}p_{0,1}}}, \quad 1 \leq j \leq n, \tag{4.2}$$

$$\frac{\tau_{p_i} - \tau_{p_{t_O}}}{\tau_{p_{t_E}} - \tau_{p_{t_O}}} = \frac{\overrightarrow{p_{0,0}q_{i,t}}}{\overrightarrow{p_{0,0}p_{0,1}}}. \tag{4.3}$$

Using Equation (4.2), Equation (4.1) is equivalent to

$$\overrightarrow{p_{0,0}p_i} = \sum_{j=1}^{n} \frac{\overrightarrow{p_{0,0}q_{i,j}}}{\overrightarrow{p_{0,0}p_{0,1}}} \overrightarrow{p_{i,0}p_{i,j}} + \overrightarrow{p_{0,0}p_{i,0}}. \tag{4.4}$$

Considering the fact that α is a linear transformation, and using equation (4.4), the following holds:

$$\alpha(\overrightarrow{p_{0,0}p_i}) = \sum_{j=1}^{n} \frac{\overrightarrow{p_{0,0}q_{i,j}}}{\overrightarrow{p_{0,0}p_{0,1}}} \alpha(\overrightarrow{p_{i,0}p_{i,j}}) + \alpha(\overrightarrow{p_{0,0}p_{i,0}}).$$

Moreover, let $e_i(\tau)$ be the ith vector of the standard spatial basis at time τ and denote by $e_i = e_i(0)$. We then have

$$\alpha(\overrightarrow{p_{0,0}p_i}) = \sum_{j=1}^{n} \frac{\overrightarrow{p_{0,0}q_{i,j}}}{\overrightarrow{p_{0,0}p_{0,1}}} \overrightarrow{e_0(\tau_i)e_j(\tau_i)} + \overrightarrow{e_0e_0(\tau_i)}. \tag{4.5}$$

As equation (4.3) is invariant under elements of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$, we also have that

$$\frac{\alpha(\tau_{p_i}) - \alpha(\tau_{p_{t_O}})}{\alpha(\tau_{p_{t_E}}) - \alpha(\tau_{p_{t_O}})} = \frac{\tau_i - 0}{1 - 0} = \tau_i = \frac{\overrightarrow{p_{0,0}q_{i,t}}}{\overrightarrow{p_{0,0}p_{0,1}}}. \tag{4.6}$$

So we have that:

$$\alpha(\overrightarrow{p_{0,0}p_i}) = \sum_{j=1}^{n} \frac{\overrightarrow{p_{0,0}q_{i,j}}}{\overrightarrow{p_{0,0}p_{0,1}}} \overrightarrow{e_0(\tau_i)e_j(\tau_i)} + \frac{\overrightarrow{p_{0,0}q_{i,t}}}{\overrightarrow{p_{0,0}p_{0,1}}} \overrightarrow{e_0e_{n+1}}.$$

Since all standard bases $(e_0(\tau_i), \ldots, e_n(\tau_i))$ are parallel along the time axis, we have that

$$\alpha(\overrightarrow{p_{0,0}p_i}) = \sum_{j=1}^{n} \frac{\overrightarrow{p_{0,0}q_{i,j}}}{\overrightarrow{p_{0,0}p_{0,1}}} \overrightarrow{e_0e_j} + \frac{\overrightarrow{p_{0,0}q_{i,t}}}{\overrightarrow{p_{0,0}p_{0,1}}} \overrightarrow{e_0e_{n+1}}.$$

This completes the correctness proof for the conversion of relational predicates.

• *The translation of composed formulas.* When all the atomic sub-formulas of ψ have been translated as described above, the logical connectives can be added in a natural way. We assume that two atomic formulas χ_1 and χ_2 are translated already, into χ_1^{st} and χ_2^{st}. The translations of $\chi_1 \wedge \chi_2$ and $\chi_1 \vee \chi_2$ are $\chi_1^{st} \wedge \chi_2^{st}$ and $\chi_1^{st} \vee \chi_2^{st}$, respectively. The formula $\neg\chi_1$ is translated into $\neg\chi_1^{st}$.

Remember that with the conversion of a formula ψ_{rel} of type $R(x_{1,1}, \ldots, x_{1,n}, x_{1,t}, \ldots, x_{m,1}, \ldots, x_{m,n}, x_{m,t})$ we assumed that $x_{i,t} \neq x_{j,t}$ for any $(1 \leq i < j < m)$. The reason is that we want to have only one affine coordinate system for every different time moment considered in that formula. Indeed, an element α of $(\mathcal{A}_{st}^f, \mathcal{A}_t)$ is a one-to-one mapping from the snapshots of a certain input database $\mathfrak{F}$ to the snapshots of the output database $\alpha(\mathfrak{F})$. Therefore, we cannot map two different co-temporal coordinate systems to the same standard coordinate system using such an affinity.

Suppose now that $x_{i,t} = x_{j,t}$ for some $(1 \leq i < j \leq m)$. Then we adapt the previous translation with the extra requirement that $v_{i,k} = v_{j,k}$ for $k = 0, \ldots, n$ and we have unique coordinate system for each point occurring in time.

When translating an FO(+, ×, <, 0, 1, σ)-formula ψ, it is in general not known in advance which time coordinates are equal (this may depend on the input database; and it is undecidable in general which time coordinates are equal in an FO(+, ×, <, 0, 1, σ)-formula). To circumvent this problem, we consider all possible orders (using **Before**) of the time variables of ψ (a real variable denoting a time moment is recognized as it appears on the $i(n+1)$-th place $(i = 1, \ldots, m)$ in the argument list of a spatio-temporal relation predicate) and take the disjunction over all possible orders of these time variables. We denote the set of all possible orders by P.

For each $\rho \in P$ the formula ψ_ρ^{st} is the translation of ψ taken the (in)equalities into account according to the order of the time variables corresponding to ρ. Hence, each ψ_ρ^{st}-formula can have a different number ℓ_ρ of free variables, depending on ρ. We denote by ℓ the total number of free variables across all formulas ψ_ρ^{st}, $\rho \in P$.

When connecting several sub-formulas, the same principle has to be used, as arithmetic sub-formulas can impose equality on different time variables.

When applying the thus obtained translation of the quantifier-free part of ψ^{st} to a spatio-temporal database instance, only some of the disjuncts will apply (possibly depending on the particular input database).

• *Formulas with quantifiers.* Finally, the quantifier prefix of ψ is translated in the natural way. Suppose that we already translated the quantifier-free formula χ into the formula χ^{st}. Then the translation of $\exists x\,(\chi)$ is $\exists v\,(\chi^{st})$, where v is the point variable associated to x which we have already declared to be collinear with $u_{0,0}$ and $u_{0,1}$. □

Lemma 4.25 (**Completeness of** FO($\Pi(\mathcal{A}_{st}^f, \mathcal{A}_t)$, σ^{st})). Let σ^{st} be a spatio-temporal database schema. For every $(\mathcal{A}_{st}^f, \mathcal{A}_t)$-generic spatio-temporal query expressible in FO(+, ×, <, 0, 1, σ), there exists an equivalent FO($\Pi(\mathcal{A}_{st}^f, \mathcal{A}_t)$, σ^{st})-query.

Proof. Given a $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-generic spatio-temporal query of output type (n, k), expressible in FO(+, ×, <, 0, 1, σ),

$$\psi(x_{1,1}, \ldots, x_{1,n}, x_{1,t}, \ldots, x_{k,1}, \ldots, x_{k,n}, x_{k,t}).$$

The conversion procedure, given in Lemma 4.24, returns a formula

$$\psi^{st}(u_{t_O}, u_{t_E}, u_{0,0}, \ldots, u_{0,n}, u_{1,0}, \ldots, u_{1,n}, \ldots, u_{k,0}, \ldots, u_{k,n}, \\ v_{1,1}, \ldots, v_{1,n}, v_{1,t}, \ldots, v_{k,1}, \ldots, v_{k,n}, v_{k,t}),$$

parameterized by one temporal and k spatial coordinate systems and which is, up to a transformation of the group $(\mathcal{A}^f_{st}, \mathcal{A}_t)$, that depends on the coordinate systems, equivalent to the original formula ψ. Since it has additional free variables, the query ψ^{st} clearly has the wrong output type. A FO($\Pi(\mathcal{A}^f_{st}, \mathcal{A}_t)$, σ^{st})-query equivalent to ψ should be a formula

$$\psi^{st}_{\text{final}}(v_1, v_2, \ldots, v_k)$$

having k free variables only. We obtain the desired formula by introducing k new point variables v_i, and for each $1 \leq i \leq k$, n new point variables $v'_{i,1}, \ldots, v'_{i,n}$ such that $v'_{i,j}$ is collinear with $u_{i,0}$ and $u_{i,1}$ and

$$\mathbf{Coordinates}^n(u_{i,0}, \ldots, , u_{i,n}, v'_{i,1}, \ldots, v'_{i,n}, v_i). \tag{1}$$

Moreover, we require that

$$\mathbf{EqCr}^{ST}(u_{0,0}, u_{0,1}, v_{i,t}, u_{t_O}, u_{t_E}, v_i) \tag{2}$$

and

$$\bigwedge_{j=1}^{n} \mathbf{EqCR^s}(u_{0,0}, u_{0,1}, v_{i,j}, u_{i,0}, u_{i,1}, v'_{i,j}). \tag{3}$$

The final formula ψ^{st}_{final} is now obtained by existentially quantifying all point variables, except for $v_1, \ldots, v_k$ in the conjunction of ψ^{st} with the expressions (1), (2) and (3).

Now consider the (partial) output of ψ^{st}_{final} when we choose a specific coordinate system for each set of variables $u_{i,0}, \ldots, u_{i,n}$. By similar reasoning as in Lemma 4.24, we obtain that this partial output equals

$$\alpha'^{-1}(\psi(\alpha(canDB(\mathfrak{F})))))$$

where $\alpha' = (\alpha'_{st}, \alpha'_t)$ and $\alpha = (\alpha_{st}, \alpha_t)$ both are transformations as specified in the statement of Lemma 4.24. This means that they both satisfy the same set of constraints, *i.e.*, $\alpha'_t = \alpha_t$ and for certain time moments τ, $\alpha'_{st}(\tau) = \alpha_{st}(\tau)$. In between those time moments α'_{st} and α_{st} can differ. However, it follows from Lemma 4.24 that $\psi(\alpha(canDB(\mathfrak{F}))) = \psi(\alpha'(canDB(\mathfrak{F})))$, for any two transformations α and α' satisfying the constraints as described in the statement of Lemma 4.24. Hence, we can conclude without loss of generality that the partial output of ψ^{st}_{final} when we fill in a specific coordinate system for each set of variables $u_{i,0}, \ldots, u_{i,n}$ equals $\alpha^{-1}(\psi(\alpha(canDB(\mathfrak{F}))))$ where α is a transformation as specified in the statement of Lemma 4.24.

If we now consider all possible coordinate systems for each set of variables $u_{i,0}$, $\ldots, u_{i,n}$

$$\psi^{st}_{\text{final}}(\mathfrak{F}) = \bigcup_c \bigcup_{\alpha_c} (\alpha_c^{-1}(\psi(\alpha_c(canDB(\mathfrak{F}))))),$$

where c ranges over all possible coordinate system assignments and α_c ranges over all transformations satisfying the constraints following from this choice of coordinate systems.

The union $\bigcup_c \bigcup_{\alpha_c} (\alpha_c^{-1}(\psi(\alpha_c(canDB(\mathfrak{F})))))$ is in fact the union over all elements α of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ of $\alpha^{-1}(\psi(\alpha(canDB(\mathfrak{F}))))$. So,

$$\psi^{st}_{\text{final}}(\mathfrak{F}) = \bigcup_{\alpha} (\alpha^{-1}(\psi(\alpha(canDB(\mathfrak{F}))))),$$

where α ranges over all elements of $(\mathcal{A}^f_{st}, \mathcal{A}_t)$.

Since ψ^{st}_{final} is a $(\mathcal{A}^f_{st}, \mathcal{A}_t)$-generic query and the group $(\mathcal{A}^f_{st}, \mathcal{A}_t)$ is semi-algebraic (we give a precise definition in Section 4.3.3), we have that for every α

$$\alpha^{-1}(\psi(\alpha(canDB(\mathfrak{F})))) = \psi(canDB(\mathfrak{F})).$$

So, finally,

$$\psi^{st}_{\text{final}}(\mathfrak{F}) = \psi(canDB(\mathfrak{F})).$$

□

Proof of Theorem 4.22. Lemma 4.23, Lemma 4.24 and Lemma 4.25 together prove Theorem 4.22. □

We are now ready to prove Proposition 4.21:

Proof of Proposition 4.21. Note that we only consider a finite number of moments in time in the proof of Lemma 4.24 (there are only a finite number of time variables in any FO(+, ×, <, 0, 1, σ)-formula φ). This implies that the transformation groups $\mathcal{A}^f_{st}$ and $\mathcal{A}_{st}$ yield the same results. So, we can use the proof given above for the group $(\mathcal{A}_{st}, \mathcal{A}_t)$. Indeed, in between the moments of time that are considered, it is indeed not important which transformation function is used. □

Theorem 4.22 has a number of corollaries. We need one more point predicates, namely $\mathbf{Pos}^n$ before we can state those corollaries. The expression $\mathbf{Pos}^n(p_0, p_1, \ldots, p_n)$ is true for $(n+1)$ co-temporal points $p_0, p_1, \ldots, p_n$ if and only if $(p_0, p_1, \ldots, p_n)$ forms a positively oriented coordinate system.

Corollary 4.26. Let σ^{st} be a spatio-temporal database schema. Let $(\mathcal{F}_{st}, \mathcal{F}_t)$ and $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ be taken from Table 4.3. The language FO($\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$, σ^{st}) is sound and complete for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic spatio-temporal queries that are expressible in FO(+, ×, <, 0, 1, σ).

Proof. It follows directly from the proof of Theorem 4.22 that, for each subgroup $(\mathcal{F}^{(f)}_{st}, \mathcal{F}_t)$ of $(\mathcal{A}^{(f)}_{st}, \mathcal{A}_t)$, the language FO($\Pi(\mathcal{F}^{(f)}_{st}, \mathcal{F}_t)$, σ^{st}) is sound and complete for the $(\mathcal{F}^{(f)}_{st}, \mathcal{F}_t)-$generic queries expressible in FO(+, ×, <, 0, 1, σ) if and only if the following three conditions are satisfied:

$(\mathcal{F}_{st}, \mathcal{F}_t)$	Sets of point predicates $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$
$(\mathcal{A}_{st}^{(f)}, \mathcal{A}_t)$	$\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$
$(\mathcal{A}_{st}^{(f)}, \mathcal{T}_t)$	$\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}, \mathbf{UnitTime}\}$
$(\mathcal{A}_{st}^{(f)}, Id_t)$	$\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}, \mathbf{UnitTime}, \mathbf{0_t}, \mathbf{1_t}\}$
$(\mathcal{S}_{st}^{(f)}, \mathcal{F}_t)$	$\Pi(\mathcal{A}_{st}^{(f)}, \mathcal{F}_t) \cup \{\mathbf{EqDist}^{st}\}$
$(\mathcal{I}_{st}^{(f)}, \mathcal{F}_t)$	$\Pi(\mathcal{A}_{st}^{(f)}, \mathcal{F}_t) \cup \{\mathbf{EqDist}^{st}, \mathbf{UnitDist}^{st}\}$
$(\mathcal{T}_{st}^{(f)}, \mathcal{F}_t)$	$\Pi(\mathcal{I}_{st}^{(f)}, \mathcal{F}_t) \cup \{\mathbf{Smaller}_i (1 \leq i \leq n), \mathbf{Pos}^n\}$

Table 4.3: An overview of the different sets of point predicate for some transformation groups. We have $\mathcal{F}_t \in \{\mathcal{A}_t, \mathcal{T}_t, Id_t\}$.

(i) the set $\Pi(\mathcal{F}_{st}^{(f)}, \mathcal{F}_t)$ contains the point predicates $\mathbf{Between}^{\mathrm{Cotemp}}$, **Before** and $\mathbf{EqCr}^{ST}$;

(ii) all elements of $\Pi(\mathcal{F}_{st}^{(f)}, \mathcal{F}_t)$ are FO(+, $\times$, $<$, 0, 1, σ)-expressible and invariant under the transformations of $(\mathcal{F}_{st}^{(f)}, \mathcal{F}_t)$;

(iii) the facts "$(v_0, v_1, \ldots, v_n)$ is the image of the standard coordinate system in the hyperplane co-temporal with v_{t_O} under an element of $(\mathcal{F}_{st}^{(f)}, \mathcal{F}_t)$" and "$(v_{t_O}, v_{t_E})$ is the image of the standard temporal coordinate system under an element of $(\mathcal{F}_{st}^{(f)}, \mathcal{F}_t)$", where $v_0, v_1, \ldots, v_n, v_{t_O}$ and v_{t_E} are points in $(n+1)$ dimensional real space, are expressible in FO($\Pi(\mathcal{F}_{st}^{(f)}, \mathcal{F}_t)$, σ^{st}).

All groups listed in Table 4.3 are subgroups of $(\mathcal{A}_{st}^{(f)}, \mathcal{A}_t)$ and satisfy the first condition. It is also straightforward to verify that they satisfy the second condition.

For the third condition, we list for every group mentioned in Table 4.3 the expressions for the spatial and temporal coordinate system. The proof that these expressions are correct are straightforward.

• *For the group* $(\mathcal{A}_{st}^{(f)}, \mathcal{A}_t)$*,* the expressions for $\mathbf{TCoSys}_{\mathcal{A}}(u_1, u_2)$ and $\mathbf{CoSys}_{\mathcal{A}}^n(u_0, u_1, \ldots, u_n)$ were given in Lemma 4.24.

• *For the group* $(\mathcal{A}_{st}^{(f)}, \mathcal{I}_t)$*,* the expression for the spatial coordinate system does not change, but

$$\mathbf{TCoSys}_{\mathcal{T}}(u_1, u_2) := \mathbf{TCoSys}_{\mathcal{A}}(u_1, u_2) \wedge \mathbf{UnitTime}(u_1, u_2).$$

• *For the group* $(\mathcal{A}_{st}^{(f)}, Id_t)$*,* the expression for the spatial coordinate system does again not change, but

$$\mathbf{TCoSys}_{Id}(u_1, u_2) := \mathbf{TCoSys}_{\mathcal{T}}(u_1, u_2) \wedge \mathbf{0_t}(u_1) \wedge \mathbf{1_t}(u_2).$$

For the following groups, we only list the expression for the spatial coordinate system. The temporal coordinate system depends on the groups $\mathcal{F}_t$ and is completely analogous to the previous cases.

- *For the group* $(\mathcal{S}_{st}^{(f)}, \mathcal{F}_t)$, we have

$$\mathbf{CoSys}_{\mathcal{S}}(u_0, u_1, \ldots, u_n) := \mathbf{CoSys}^n_{\mathcal{A}}(u_0, u_1, \ldots, u_n) \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \mathbf{EqDist}^{st}(u_0, u_i, u_0, u_j).$$

- *For the group* $(\mathcal{I}_{st}^{(f)}, \mathcal{F}_t)$, we have

$$\mathbf{CoSys}_{\mathcal{I}}(u_0, u_1, \ldots, u_n) := \mathbf{CoSys}_{\mathcal{S}}(u_0, u_1, \ldots, u_n) \wedge \bigwedge_{i=1}^{n} \mathbf{UnitDist}^{st}(u_0, u_i).$$

- *For the group* $(\mathcal{T}_{st}^{(f)}, \mathcal{F}_t)$, we have

$$\mathbf{CoSys}_{\mathcal{T}}(u_0, u_1, \ldots, u_n) := \mathbf{CoSys}_{\mathcal{I}}(u_0, u_1, \ldots, u_n) \wedge \bigwedge_{i=1}^{n} \bigwedge_{j=1}^{n} \mathbf{Smaller}_i(u_0, u_j) \wedge \mathbf{Pos}^n(u_0, u_1, \ldots, u_n).$$

□

Next, we illustrate the languages summarized in Table 4.3 with the appropriate examples of Section 4.1.5.

Example 4.27. We give the FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$)-expression φ'_3 equivalent to the $(\mathcal{I}_{st}, \mathcal{A}_t)$-generic query of Example 4.7: *Was there a collision between car A and car B?*:

$$\varphi'_3 := \exists u\, (carA(u) \wedge carB(u)).$$

Remark that this query can be expressed without the use of the point predicates from Π. □

Example 4.28. We give the FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}, \mathbf{EqDist}^{st}, \mathbf{UnitDist}^{st}, \mathbf{Smaller}_i (1 \leq i \leq n), \mathbf{Pos}^n, \mathbf{UnitTime}, \mathbf{0_t}, \mathbf{1_t}\}$) expression φ'_4 equivalent to the $(\mathcal{T}_{st}, id_t)$-generic query of Example 4.8: *Did car A pass at 500 meters north of car B at time moment* $t = 5930$*?*

The fact that a point has time coordinate 5930 can be expressed using **UnitTime**, $\mathbf{0_t}$, and $\mathbf{1_t}$. We illustrate this with a predicate expressing the fact that a point has time coordinate 3:

$$eq3t(u) := \exists v_1\, \exists v_2\, (\mathbf{1_t}(v_1) \wedge \mathbf{Before}(v_1, v_2, \wedge)\mathbf{UnitTime}(v_1, v_2, \wedge) \mathbf{Before}(v_2, u, \wedge)\mathbf{UnitTime}(v_2, u,)).$$

It can be expressed using $\mathbf{UnitDist}^{st}$ that the distance between two points is 500 (in a way comparable to the construction of the predicate $3sec$ of Example 4.19).

Now we give the expression φ'_4:

$$\exists u\, \exists v\, \exists w\, (carA(u) \wedge carB(v) \wedge eq5930t(u) \wedge eq5930t(v) \wedge (\mathbf{Smaller}_1(u, w) \wedge \mathbf{Smaller}_1(w, u)) \wedge (\mathbf{Smaller}_2(v, w) \wedge \mathbf{Smaller}_2(w, v)) \wedge 500meters(u, w)).$$

□

4.3 Computable Generic Spatio-temporal Queries

In this section, we show that the languages FO($\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$, σ^{st}) of the previous section, when extended with assignment statements and a While loop, yield languages that are computationally sound and complete for the computable queries that are $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic. To start with, we explain in more detail how point-based logics are extended with assignment statements and a while loop. Afterwards, this section is organized in the same way as Section 4.2. We first discuss sound and complete languages for the queries generic for time-independent transformation groups. Then we focus on genericity for groups related to physical notions. Finally, we address sound and complete languages for the queries that are generic for the time-dependent transformations.

We start with extending the point-based logics described in Definition 4.11 with while loops. Remark that this is similar to the extension of FO(+, $\times$, $<$, 0, 1) with a while loop, that we defined earlier in this thesis (see Definition 3.9).

Definition 4.29 (FO(Π, σ^{st}) **+ While-program**)**.** Let Π be a finite set of point predicates, and let σ^{st} be a (spatio-temporal) database schema. Syntactically, a *program* in the language FO(Π, σ^{st}) + While is a finite sequence of *statements* and *while-loops*. It is assumed there is a sufficient supply of new relation variables, each with an appropriate arity.

(i) Each statement has the form

$$R^{st} := \{(u_1, \ldots, u_k) \mid \varphi(u_1, \ldots, u_k)\};.$$

Here, R is a relation variable with assigned arity k (the variables u_i range over $(\mathbb{R}^n \times \mathbb{R})$) and φ is a formula in FO(Π, $\sigma^{st\prime}$), where $\sigma^{st\prime}$ is the set of relation names containing the elements of σ^{st} together with the relation variables introduced in previous statements of the program.

(ii) A while-loop has the form

while φ **do**
  P
end while

where P is a program and φ is a sentence in FO(Π, $\sigma^{st\prime}$), where $\sigma^{st\prime}$ is again the set of relation names containing the elements of σ^{st} together with the relation variables introduced in previous statements of the program.

(iii) One of the relation names occurring in the program is designated as the output relation and is named R^{st}_{out}.

Semantically, a program in the query language FO(Π, σ^{st}) expresses a spatio-temporal query as soon as R^{st}_{out} is assigned a return value. The execution of a FO(Π, σ^{st}) + While-program applied to an input database is performed step-by-step. A statement is executed by first evaluating the FO(Π, σ^{st})-formula on the right hand side on the input database together with the newly created relations resulting from previous statements. Next, the result of the evaluation of the right hand side is

assigned to the relation variable on the left-hand side. The effect of a while loop is to execute the body as long as the condition φ evaluates to true.

Note that a FO(Π, σ^{st}) + While-program is not guaranteed to halt. For those input databases it does not, the query represented by the program is not defined on that particular input database.

Consider the following example which will be used later on to express the query from Example 4.5.

Example 4.30. Suppose that we have a spatio-temporal database over the schema $\sigma^{st} = \{RST, S^{st}\}$, where the underlying dimension is two and both R^{st} and S^{st} have arity one. We assume that all points in R^{st} and S^{st} have disjoint time coordinates. This means that we can sort all points according to their time coordinates. We also assume that R^{st} and S^{st} both contain a finite number of points.

The query Q we want to answer is the following: *Does R^{st} contain more points than S^{st}?*. It is well known that we cannot express this query in first-order logic [37]. The FO(Π, σ^{st}) + While-program expressing Q is:

$R^{st}_{\mathrm{Not}} := \{\}$;
$S^{st}_{\mathrm{Not}} := \{\}$;
$R^{st}_{\mathrm{Smallest}} :=$

$$\{(u) \mid R^{st}(u) \wedge \neg R^{st}_{\mathrm{Not}}(u) \wedge \forall v\,((R^{st}(v) \wedge \neg R^{st}_{\mathrm{Not}}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

$S^{st}_{\mathrm{Smallest}} :=$

$$\{(u) \mid S^{st}(u) \wedge \neg S^{st}_{\mathrm{Not}}(u) \wedge \forall v\,((S^{st}(v) \wedge \neg S^{st}_{\mathrm{Not}}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

while $\exists u\,(R^{st}_{\mathrm{Smallest}}(u)) \wedge \exists v\,(S^{st}_{\mathrm{Smallest}}(v))$ **do**
$R^{st}_{\mathrm{Not}} := \{(u) \mid R^{st}_{\mathrm{Not}}(u) \vee R^{st}_{\mathrm{Smallest}}(u)\}$;
$S^{st}_{\mathrm{Not}} := \{(u) \mid S^{st}_{\mathrm{Not}}(u) \vee S^{st}_{\mathrm{Smallest}}(u)\}$;
$R^{st}_{\mathrm{Smallest}} :=$

$$\{(u) \mid R^{st}(u) \wedge \neg R^{st}_{\mathrm{Not}}(u) \wedge \forall v\,((R^{st}(v) \wedge \neg R^{st}_{\mathrm{Not}}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

$S^{st}_{\mathrm{Smallest}} :=$

$$\{(u) \mid S^{st}(u) \wedge \neg S^{st}_{\mathrm{Not}}(u) \wedge \forall v\,((S^{st}(v) \wedge \neg S^{st}_{\mathrm{Not}}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

end while
$R^{st}_{\mathrm{out}} := \{() \mid \exists u\,(R^{st}_{\mathrm{Smallest}}(u))\}$;

Intuitively, this program repeatedly takes the earliest point from both R^{st} and S^{st} until they do not both contain unvisited points anymore. When the while loop terminates and R^{st} still contains unvisited points, true is returned. □

4.3.1 Genericity for Time-independent Transformations

In this section, we prove a general result concerning computable $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic queries where $(\mathcal{F}_{st}, \mathcal{F}_t)$ is a time-independent affinity of $(\mathbb{R}^n \times \mathbb{R})$, *i.e.*, a group from Table 4.1. The following theorem follows directly from the proof of Theorem 6.1 [43].

Theorem 4.31 (**Computable meta-theorem**)**.** Let σ^{st} be a spatio-temporal database schema, let $\mathcal{F}$ be a subgroup of the affinities of $(\mathbb{R}^n \times \mathbb{R})$, let Π be a set of point predicates and let FO(Π, σ^{st}) be a point language that is sound and complete for the $\mathcal{F}$-generic queries expressible in FO($+$, $\times$, $<$, 0, 1, σ). Then the language FO(Π, σ^{st}) + While is sound and complete for the $\mathcal{F}$-generic computable queries.

From this, we can derive the following result:

Corollary 4.32 (**Complete languages for time-independent groups**)**.** Let σ be a spatio-temporal database schema. Let $(\mathcal{F}_{st}, \mathcal{F}_t)$ be a group and let $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ be as in Table 4.1. The point language FO($\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$, σ^{st}) + While is sound and complete for the computable $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic queries.

Proof. The correctness follows from Theorem 4.16 and Theorem 4.31. □

4.3.2 Applications to Physics

Here, we focus again on the transformation groups $(\mathcal{V}_{st}, \mathcal{T}_t)$, $(\mathcal{V}(\mathcal{R})_{st}, \mathcal{T}_t)$, $(\mathcal{AC}_{st}, \mathcal{T}_t)$ and $(\mathcal{AC}(\mathcal{R})_{st}, \mathcal{T}_t)$. As they are all subgroups of the affinities of $(\mathbb{R}^n \times \mathbb{R})$, we can apply Theorem 4.31 again.

Corollary 4.33 (**Complete languages for transformations from physics**)**.** Let σ^{st} be a spatio-temporal database schema. Let $(\mathcal{F}_{st}, \mathcal{T}_t)$ be a group from Table 4.2 and let $\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$ be as in Table 4.2. The point language FO($\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$, σ^{st}) + While is sound and complete for the $(\mathcal{F}_{st}, \mathcal{I}_t)$-generic computable spatio-temporal queries.

Proof. The correctness follows from Theorem 4.18 and Theorem 4.31. □

Example 4.34. We now give the FO($\{\mathbf{Between}^{(n+1)}, \mathbf{Before}, \mathbf{EqSpace}\}$) + While-program expressing query Q_1 of Example 4.5: *Does the route followed by car A self-intersect more often than the route followed by car B does?*.

If a car is standing still at a certain position, this will result in an infinite number of points in $(\mathbb{R}^n \times \mathbb{R})$ with the same spatial coordinates. However, one would not consider this situation to be an infinite number of self-intersections. Therefore, when such a situation happens, we only consider the last moment of the interval during which the car is at that specific location.

Intuitively, the program first computes the relations containing all self-intersections of the trajectories of both cars, and then determines whether the route of car A self-intersects the most. The program of Example 4.30 can be used to perform this last task. We slightly adapt it such that it expresses query Q_1:

□

$A_{\cap} :=$

$$\{(u) \mid carA(u) \wedge \exists v\,(carA(v) \wedge \mathbf{Before}(u,v) \wedge \mathbf{EqSpace}(u,v) \wedge \forall w\,((carA(w) \wedge \mathbf{Before}(u,w) \wedge \mathbf{Before}(w,v) \wedge u \neq w \wedge v \neq w) \rightarrow \neg(\mathbf{EqSpace}(w,v))))\};$$

$B_{\cap} :=$

$$\{(u) \mid carB(u) \wedge \exists v\,(carB(v) \wedge \mathbf{Before}(u,v) \wedge \mathbf{EqSpace}(u,v) \wedge \forall w\,((carB(w) \wedge \mathbf{Before}(u,w) \wedge \mathbf{Before}(w,v) \wedge u \neq w \wedge v \neq w) \rightarrow \neg(\mathbf{EqSpace}(w,v))))\};$$

$A_{\mathrm{N}ot} := \{\};$
$B_{\mathrm{N}ot} := \{\};$
$A_{\mathrm{S}mallest} :=$

$$\{(u) \mid A_{\cap}(u) \wedge \neg A_{\mathrm{N}ot}(u) \wedge \forall v\,((A_{\cap}(v) \wedge \neg A_{\mathrm{N}ot}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

$B_{\mathrm{S}mallest} :=$

$$\{(u) \mid B_{\cap}(u) \wedge \neg B_{\mathrm{N}ot}(u) \wedge \forall v\,((B_{\cap}(v) \wedge \neg B_{\mathrm{N}ot}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

while $\exists u\,(A_{\mathrm{S}mallest}(u)) \wedge \exists v\,(B_{\mathrm{S}mallest}(v))$ **do**
  $A_{\mathrm{N}ot} := \{(u) \mid A_{\mathrm{N}ot}(u) \vee A_{\mathrm{S}mallest}(u)\};$
  $B_{\mathrm{N}ot} := \{(u) \mid B_{\mathrm{N}ot}(u) \vee B_{\mathrm{S}mallest}(u)\};$
  $A_{\mathrm{S}mallest} :=$

$$\{(u) \mid A_{\cap}(u) \wedge \neg A_{\mathrm{N}ot}(u) \wedge \forall v\,((A_{\cap}(v) \wedge \neg A_{\mathrm{N}ot}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

  $B_{\mathrm{S}mallest} :=$

$$\{(u) \mid B_{\cap}(u) \wedge \neg B_{\mathrm{N}ot}(u) \wedge \forall v\,((B(v) \wedge \neg B_{\mathrm{N}ot}(v)) \rightarrow (\mathbf{Before}(u,v)))\};$$

end while
$R_{\mathrm{out}} := \{() \mid \exists u\,(A_{\mathrm{S}mallest}(u))\};$

4.3.3 Genericity for Time-dependent Transformations

Finally, we study notions of genericity determined by groups of time-dependent transformations. Here, we only show results for the groups of arbitrary time-dependent transformations $\mathcal{F}_{st}$. We concentrate on the group $(\mathcal{A}_{st}, \mathcal{A}_t)$. The other time-dependent transformation groups will be addressed afterwards (see Corollary 4.40). For the groups $\mathcal{F}^f_{st}$ the problem of identifying sound and complete languages is open, we will discuss the problems concerning this at the end of this section.

We introduce some definitions first. Recall that we introduced, in Section 4.1.1, the abbreviation $f(R^{\mathfrak{F}})$ for the formula $\{(f(\mathbf{a}_1, \tau_1), f(\mathbf{a}_2, \tau_2), \ldots, f(\mathbf{a}_k, \tau_k)) \mid (\mathbf{a}_1, \tau_1, \mathbf{a}_2, \tau_2, \ldots, \mathbf{a}_k, \tau_k) \in R^{st\mathfrak{F}}\}$, where R^{st} is a relation name and $\mathfrak{F}$ a spatio-temporal database over a schema σ^{st} that contains R^{st}.

Definition 4.35 (**$(\mathcal{F}_{st}, \mathcal{F}_t)$-isomorphic databases**)**.** Let $\mathfrak{F}_1$ and $\mathfrak{F}_2$ be spatio-temporal databases over the schema $\sigma^{st} = \{R^{st}_1, \ldots, R^{st}_m\}$ with underlying dimension n. The databases $\mathfrak{F}_1$ and $\mathfrak{F}_2$ are called $(\mathcal{F}_{st}, \mathcal{F}_t)$-*isomorphic* if and only if there exists a $f = (f_{st}, f_t) \in (\mathcal{F}_{st}, \mathcal{F}_t)$ such that for all R^{st}_i in σ^{st}, $f(R^{st\mathfrak{F}_1}_i) = R^{st\mathfrak{F}_2}_i$.

Recall that a representation of a spatio-temporal database $\mathfrak{F}$ over a schema $\sigma^{st} = \{R^{st}_1, \ldots, R^{st}_m\}$ is a tuple $(\varphi_1, \ldots, \varphi_m)$ of quantifier-free formulas in FO(+, ×, <, 0, 1), such that φ_i describes $R^{st\mathfrak{F}}_i$.

Assuming some order on the characters or symbols that may appear in a FO(+, ×, <, 0, 1)-formulas, we can lexicographically order the FO(+, ×, <, 0, 1)-formulas.

Definition 4.36 (**$(\mathcal{F}_{st}, \mathcal{F}_t)$-canonization**)**.** The $(\mathcal{F}_{st}, \mathcal{F}_t)$-*canonization* of a spatio-temporal database $\mathfrak{F}$ over a schema $\sigma^{st} = \{R^{st}_1, \ldots, R^{st}_m\}$, denoted $\mathbf{Canon}_{(\mathcal{F}_{st}, \mathcal{F}_t)}(\mathfrak{F})$, is the spatio-temporal database $\mathfrak{F}'$, which is $(\mathcal{F}_{st}, \mathcal{F}_t)$-isomorphic to $\mathfrak{F}$ and has a representation by quantifier-free FO(+, ×, <, 0, 1)-formulas

$$(\varphi_{\mathbf{Canon}_{(\mathcal{F}_{st}, \mathcal{F}_t)}(R^{st}_1)}, \ldots, \varphi_{\mathbf{Canon}_{(\mathcal{F}_{st}, \mathcal{F}_t)}(R^{st}_m)})$$

that occurs lexicographically first among the representations of spatio-temporal databases $(\mathcal{F}_{st}, \mathcal{F}_t)$-isomorphic to $\mathfrak{F}$.

Definition 4.37 (**$(\mathcal{F}_{st}, \mathcal{F}_t)$-type**)**.** Let $\mathfrak{F}$ be a spatio-temporal database. The $(\mathcal{F}_{st}, \mathcal{F}_t)$-*type of* $\mathfrak{F}$, denoted $Type_{(\mathcal{F}_{st}, \mathcal{F}_t)}(\mathfrak{F})$, equals

$$\{f \in (\mathcal{F}_{st}, \mathcal{F}_t) \mid f(\mathfrak{F}) = \mathbf{Canon}_{(\mathcal{F}_{st}, \mathcal{F}_t)}(\mathfrak{F})\}.$$

We can derive directly from a similar proposition of Gyssens, Van den Bussche and Van Gucht [43] that, for a spatio-temporal database $\mathfrak{F}$, a representation of $\mathbf{Canon}_{(\mathcal{F}_{st}, \mathcal{F}_t)}(\mathfrak{F})$ can be computed if and only if $(\mathcal{F}_{st}, \mathcal{F}_t)$ is a *transformation group*.

A transformation group $\mathcal{G}$ of $(\mathbb{R}^n \times \mathbb{R})$ is semi-algebraic if and only if there exists a semi-algebraic subset of $\mathbb{R}^\ell$, described by a FO(+, ×, <, 0, 1)-formula $\varphi_{\mathcal{G}}$, for some fixed ℓ, representing all elements of $\mathcal{G}$, such that the set

$$\{(g_1, \ldots, g_\ell, x_1, \ldots, x_n, t, x'_1, \ldots, x'_n, t') \mid \varphi_{\mathcal{G}}(g_1, \ldots, g_\ell) \wedge \varphi_{\mathcal{G}-img}(g_1, \ldots, g_\ell, x_1, \ldots, x_n, t, x'_1, \ldots, x'_n, t')\},$$

also called the *graph* of $\mathcal{G}$, is a semi algebraic subset of $\mathbb{R}^{\ell+2(n+1)}$. The formula $\varphi_{\mathcal{G}-img}$ expresses that, for the element of $\mathcal{G}$ represented by the tuple $(g_1, \ldots, g_\ell)$, the tuple $(x_1, \ldots, x_n, t)$ is mapped to $(x'_1, \ldots, x'_n, t')$.

Remark 4.38. The transformation group $(\mathcal{A}_{st}, \mathcal{A}_t)$ is semi-algebraic if and only if the transformation groups $\mathcal{A}_{st}$ and $\mathcal{A}_t$ are both semi-algebraic. The semi-algebraic set for the group $\mathcal{A}_t$ is $\varphi_{\mathcal{A}_t} \equiv \{(\alpha, \beta) \in \mathbb{R}^2 \mid \alpha > 0\}$. The semi-algebraic set associated to the group $\mathcal{A}_{st}$ is given by the FO(+, $\times$, <, 0, 1)-formula $\varphi_{\mathcal{A}_{st}}(\alpha_{1,1}, \ldots, \alpha_{1,n}, \ldots, \alpha_{n,1}, \ldots, \alpha_{n,n}, \beta_1, \ldots, \beta_n, t)$ expressed by

$$\begin{vmatrix} \alpha_{1,1} & \cdots & \alpha_{1,n} \\ \vdots & \cdots & \vdots \\ \alpha_{n,1} & \cdots & \alpha_{n,n} \end{vmatrix} \neq 0.$$

We now give the formulas for $\varphi_{\mathcal{A}_t-img}$ and $\varphi_{\mathcal{A}_{st}-img}$. The formula $\varphi_{\mathcal{A}_t-img}(\alpha, \beta, x_1, \ldots, x_n, x_t, x'_1, \ldots, x'_n, x'_t)$ can be given as

$$x'_t = \alpha x_t + \beta \wedge \bigwedge_{i=1}^{n} x'_i = x_i.$$

The formula $\varphi_{\mathcal{A}_{st}-img}(\alpha_{1,1}, \ldots, \alpha_{1,n}, \ldots, \alpha_{n,1}, \ldots, \alpha_{n,n},\ \beta_1, \ldots, \beta_n, t,\ x_1, \ldots, x_n, x_t, x'_1, \ldots, x'_n, x'_t)$ on the other hand is

$$t = t_x \wedge t_x = t'_x \wedge \bigwedge_{i=1}^{n} \alpha_{i,1}x_1 + \cdots + \alpha_{i,n}x_n + \beta_i = x'_i.$$

The graph of the group $(\mathcal{A}_{st}, \mathcal{A}_t)$ is now a semi-algebraic subset of $\mathbb{R}^{\ell+2(n+1)}$, where $\ell = n^2 + n + 3$, described as follows:

$$\begin{aligned} \gamma_{(\mathcal{A}_{st},\mathcal{A}_t)}(&\alpha_{1,1}, \ldots, \alpha_{1,n}, \ldots, \alpha_{n,1}, \ldots, \alpha_{n,n}, \beta_1, \ldots, \beta_n, t, \\ &\alpha, \beta, x_1, \ldots, x_n, t_x, x'_1, \ldots, x'_n, t'_x) := \\ &\varphi_{\mathcal{A}_{st}-img}(\alpha_{1,1}, \ldots, \alpha_{1,n}, \ldots, \alpha_{n,1}, \ldots, \alpha_{n,n}, \beta_1, \ldots, \beta_n, t, \\ &\quad x_1, \ldots, x_n, x_t, x'_1, \ldots, x'_n, x_t) \\ &\quad \wedge \varphi_{\mathcal{A}_t-img}(\alpha, \beta, x'_1, \ldots, x'_n, x_t, x'_1, \ldots, x'_n, x'_t). \end{aligned}$$

We now prove the main theorem of this section. The proof technique used here was introduced by Gyssens, Van den Bussche and Van Gucht [43]. We first sketch the proof technique, but only give details about the aspects of the proof that need modifications in the context of spatio-temporal databases. These modifications are based on proof techniques introduced in Section 4.2.

Theorem 4.39 (Computational completeness of FO($\Pi(\mathcal{A}_{st}, \mathcal{A}_t)$, σ^{st}) +While). Let σ^{st} be a spatio-temporal database schema. The point language FO($\Pi(\mathcal{A}_{st}, \mathcal{A}_t)$, σ^{st}) +While is sound and complete for the $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic computable spatio-temporal queries.

Proof. It suffices to show that any $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic computable query Q over σ can be simulated in the language FO($\Pi(\mathcal{A}_{st}, \mathcal{A}_t)$, σ^{st}) + While. We first briefly sketch the proof strategy, including the conversion procedure and the encoding and decoding step, that appear in it. Later the coding and decoding will be explained in more detail. For the remainder of this proof, we will abbreviate $\Pi(\mathcal{A}_{st}, \mathcal{A}_t)$ by Π.

We start with the encoding that will be used to convert formulas that represent spatio-temporal relations into natural numbers

The encoding mechanism.

Let $\mathfrak{F}$ be a spatio-temporal database over σ^{st}. Let K be the maximum of the arities of all relations in σ^{st} and the query Q. Let n be the underlying dimension. Then each relation of $\mathfrak{F}$ can be represented by a quantifier-free FO(+, $\times$, $<$, 0, 1)-formula using only the variables $x_1, \ldots, x_{(n+1)K}$, the symbols $\leq, +, \times, (,), \vee$ and $\neg$, and the constants 0 and 1.

We denote these $9 + (n + 1)K$ by $s_1, \ldots, s_{9+(n+1)K}$. Hence, we can encode a quantifier-free FO(+, $\times$, $<$, 0, 1)-formula as a string $s = s_{i_1} \ldots s_{i_k}$ and that string as the natural number $N = p_1^{i_1} \ldots p_k^{i_k}$, where p_j is the j-th prime number. We denote N by $Encode(s)$.

Proof strategy.

Given a spatio-temporal database $\mathfrak{F}$ over a schema $\sigma^{st} = \{R_1^{st}, \ldots, R_m^{st}\}$, the simulation of a $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic k-ary computable query on input $\mathfrak{F}$ is broken up into three steps:

(i) *The encoding step:* The database $\mathfrak{F}$ is encoded as a tuple of natural numbers $(N_{R_1^{st}}, \ldots, N_{R_m^{st}})$, one for each relation of the database. Here, $N_{R_i^{st}} = Encode(s_i)$, where $(s_1, \ldots, s_m)$ are the string representation of the quantifier-free formulas $\varphi_{\mathbf{Canon}_{(\mathcal{A}_{st}, \mathcal{A}_t)}(R_i^{st})}$ $(i = 1, \ldots, m)$ of the database $\mathbf{Canon}_{(\mathcal{A}_{st}, \mathcal{A}_t)}(\mathfrak{F})$. It will be shown below that this encoding can be performed in the language FO(Π, σ^{st}) + While. The set $Type_{(\mathcal{A}_{st}, \mathcal{A}_t)}(\mathfrak{F})$ is also computed, to be used in the decoding step.

(ii) *The computing step:* It can be shown easily that the language FO(Π, σ^{st}) + While has full computational power on the natural numbers, by simulating a counter machine (cfr [43]).

More specifically, one can simulate a counter machine M in FO(Π, σ^{st}) + While such that on input $(N_{R_1^{st}}, \ldots, N_{R_m^{st}})$, M halts if and only if Q is defined on the corresponding $\mathfrak{F}$ and M will output a natural number N_q which is the encoding of $Q(\mathfrak{F})$.

(iii) *The decoding step:* If M terminates on input $(N_{R_1^{st}}, \ldots, N_{R_m^{st}})$ then it outputs a natural number N_q. Using $Type_{(\mathcal{A}_{st}, \mathcal{A}_t)}(\mathfrak{F})$, the decoding algorithm computes the point set of which N_q is the encoding. This can be implemented in the language FO(Π, σ^{st}) + While.

We show next the details in the encoding and decoding algorithms that are different for $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic queries, as compared to the affine-generic queries considered

in [43]. For ease of exposition, we will assume for the remainder of this proof that the input spatio-temporal database $\mathfrak{F}$ has only one relation, with arity one, *i.e.*, $\sigma^{st} = \{R^{st}\}$. For relations with arity greater than one, the encoding algorithm has to consider more variables. If the input database contains more relations, each relation has to be encoded separately.

The encoding algorithm can be expressed in FO(Π, σ^{st}) + While.

Roughly speaking, the encoding procedure enumerates all natural numbers and meanwhile stores the evaluation of the terms and formulas that are encoded by those numbers in relations that are called T and F, respectively. This enumeration continues until one natural number is found that encodes a relation that is $(\mathcal{A}_{st}, \mathcal{A}_t)$-isomorphic to R^{st}. This relation, for which the evaluation is stored in F, corresponds to $\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(R^{st})$. The set $Type_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F})$ is also computed, to use in the decoding step.

First, we explain the role of the relations T and F in more detail, as well as the way they are built during the encoding process.

The encoding program builds up terms and formulas until the formula is found that encodes $\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F})$. The terms and formulas are stored in the relations T and F. In general, the arity of T is $(n+1)+2+2+l\times(n+1)$, where n is the underlying dimension and $l = ar(R)$. Under the assumption that $ar(R) = 1$ and the underlying dimension is 2, each tuple in T is of the form

$$(u_{t_O}, u_{t_E}, u_0, u_1, u_2, u_t, p_1, p_2, p_t, v),$$

where (u_{t_O}, u_{t_E}) is a temporal coordinate system, (u_0, u_1, u_2) a spatial coordinate system, u_t the encoding of a term which only uses the variables x_1, x_2, x_t (which are translated into v_1, v_2, v_t), and v the value of the term when evaluated under the valuation $v_1 \mapsto p_1, v_2 \mapsto p_2, v_t \mapsto p_t$. The arity of F is $(n+1)+2+1+l\times(n+1)$. Under the same assumptions, each tuple in F is of the form

$$(u_{t_O}, u_{t_E}, u_0, u_1, u_2, u_f, p_1, p_2, p_t),$$

where (u_{t_O}, u_{t_E}) and (u_0, u_1, u_2) are as before, u_f the encoding of a formula $\overline{\varphi}$ which only uses the variables x_1, x_2, x_t, and where $\varphi(p_1, p_2, p_t)$ is true.

In Algorithm 1, we give the structure of the encoding program in FO(Π, σ^{st}) + While. In this algorithm, it is assumed that substrings s' of a string s is encountered in the enumeration before s is encountered.

We will discuss in detail

(i) the representation of natural numbers (as we only have point-variables),

(ii) the expression that checks whether a certain natural number encodes a formula which represents $\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F})$, and

(iii) the computation of the set $Type_{(\mathcal{A}_{st},\mathcal{A}_t)}(R^{st})$.

All other elements of the encoding can be adopted from the proof of [43] with only slight modifications. For ease of exposition, we give the formulas for $n = 2$.

Algorithm 1 The encoding program. The input is a FO(+, ×, <, 0, 1)-sentence.

$m := 0$
$T := \emptyset$
$F := \emptyset$
found :=False
while not found **do**
  $m := m + 1$
  if m encodes x_1 **then**
    $T := T \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t, p_1) \mid p_1, p_2, p_t$ collinear with u_0 and $u_1\}$
  else if m encodes x_2 **then**
    $T := T \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t, p_2) \mid p_1, p_2, p_t$ collinear with u_0 and $u_1\}$
  else if m encodes x_t **then**
    $T := T \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t, p_t) \mid p_1, p_2, p_t$ collinear with u_0 and $u_1\}$
  else if m encodes 0 **then**
    $T := T \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t, u_0) \mid p_1, p_2, p_t$ collinear with u_0 and $u_1\}$
  else if m encodes 1 **then**
    $T := T \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t, u_1) \mid p_1, p_2, p_t$ collinear with u_0 and $u_1\}$
  else if m encodes $(s + t)$ **then**
    $T := T \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t, p_e) \mid T(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(s), p_1, p_2, p_t, p_c) \wedge T(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(t), p_1, p_2, p_t, p_d) \wedge \mathbf{Plus}(p_c, p_d, p_e)\}$
  else if m encodes $(s \times t)$ **then**
    $T := T \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t, p_e) \mid T(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(s), p_1, p_2, p_t, p_c) \wedge T(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(t), p_1, p_2, p_t, p_d) \wedge \mathbf{Times}(p_c, p_d, p_e)\}$
  else if m encodes $(s \leq t)$ **then**
    $F := F \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t) \mid \exists c \exists d\,(T(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(s), p_1, p_2, p_t, p_c) \wedge T(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(t), p_1, p_2, p_t, p_d) \wedge \mathbf{Less}(p_c, p_d))\}$
  else if m encodes $(\neg\varphi)$ **then**
    $F := F \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t) \mid \neg F(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(\varphi), p_1, p_2, p_t)\}$
  else if m encodes $(\varphi \vee \psi)$ **then**
    $F := F \cup \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t) \mid F(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(\varphi), p_1, p_2, p_t) \vee F(u_{t_O}, u_{t_E}, u_0, u_1, u_2, enc(\psi), p_1, p_2, p_t)\}$
  end if
  found:= m encodes a formula which represents $\mathbf{Canon}_{(\mathcal{A}_{st}, \mathcal{A}_t)}(R)$
end while
$N_{\mathbf{Canon}_{(\mathcal{A}_{st}, \mathcal{A}_t)}(R)} := m$
$Type_{(\mathcal{A}_{st}, \mathcal{A}_t)} := \{a \in (\mathcal{A}_{st}, \mathcal{A}_t) \mid a(R) = \mathbf{Canon}_{(\mathcal{A}_{st}, \mathcal{A}_t)}(R)\}$

(i) Natural numbers can be represented by $(n+1)$-dimensional points using the computation plane technique introduced in Section 4.2. Further on, in de encoding and decoding algorithm, we need to simulate assignments such as $m := 0$ and $m := m + 1$ (since we have to run through all natural numbers in those algorithms). As an illustration, we explain here how these are simulated in FO(Π, σ^{st}) + While. The expression $m := 0$, for example, is translated in FO(Π, σ^{st}) + While by assigning to a spatio-temporal relation a point that is the origin of the chosen computation plane. The translated expression is

$$N := \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, v) \mid \mathbf{TCoSys}_{\mathcal{A}}(u_{t_O}, u_{t_E}) \wedge \mathbf{CoSys}^n_{\mathcal{A}}(u_0, u_1, u_2) \wedge \mathbf{Collinear}(u_0, u_1, v) \wedge v = u_0\}.$$

For the assignment $m := m + 1$, we have:

$$N := \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, v) \mid \exists w\,(N(u_{t_O}, u_{t_E}, u_0, u_1, u_2, w) \wedge \mathbf{Plus}(u_{t_O}, u_{t_E}, u_0, u_1, u_2, w, u_1, v))\}.$$

The predicate **Plus**, which expresses that, relative to a computation plane, a certain point represents the sum of two other points can be written in FO(Π, σ^{st}) + While because of Theorem 4.22.

(ii) We now give the expression φ that checks whether a certain natural number m encodes a formula which represents $\varphi_{\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(R_i^{st})}$ $(i = 1, \ldots, m)$. Remember that the evaluation of the formula encoded by m is stored in the relation F. This relation has arity $n + 3 + K$, where K is the maximal arity in the input database schema. Let $(p_1, p_2, \ldots, p_{n+3+K})$ be a tuple of points satisfying F. The points p_1 and p_2 are a temporal coordinate system, p_{n+3} represents the natural number m encoding the formula and $p_3, p_4, \ldots, p_{n+2}$ form a hyperplane of which the plane through p_3, p_4 and p_5 will be use as a computation plane. The last K points are the translation of the free variables in the formula encoded by m.

Let the formula $\psi_{(\mathcal{A}_{st},\mathcal{A}_t)}$ be the translation of the formula $\varphi_{(\mathcal{A}_{st},\mathcal{A}_t)}$ from Example 4.38. Intuitively, the next formula checks, for a natural number m, whether there exists an element of the group $(\mathcal{A}_{st}, \mathcal{A}_t)$ that maps each point in R^{st} to a point in the set of points satisfying the formula encoded by m, the evaluation of which is stored in F.

The following formula ψ checks whether the right quantifier-free formula has been found. It reflects the stop condition of the While-loop that runs through the natural numbers. This formula ψ can be written as

$$\forall u_{t_O}\,\forall u_{t_E}\,\forall u_0\,\forall u_1\,\forall u_2\,(\mathbf{TCoSys}_{\mathcal{A}}(u_{t_O},u_{t_E})\wedge$$
$$\mathbf{CoSys}^n_{\mathcal{A}}(u_0,u_1,u_2))\rightarrow\exists v_\alpha\,\exists v_\beta\,\exists w\,\forall u_t\,\exists v_{a_{1,1}}\,\exists v_{a_{1,2}}\,\exists v_{a_{2,1}}\,\exists v_{a_{2,2}}$$
$$\exists v_{b_1}\,\exists v_{b_2}\,(N(u_{t_O},u_{t_E},u_0,u_1,u_2,w)\wedge$$
$$\forall v_x\,\forall v_y\,\forall v_t\,(F(u_{t_O},u_{t_E},u_0,u_1,u_2,w,v_x,v_y,v_t)\leftrightarrow$$
$$\exists v\,\exists v'_x\,\exists v'_y\,(R^{st}(v)\wedge\mathbf{comp\text{-}coord}(u_{t_O},u_{t_E},u_0,u_1,u_2,v,v'_x,v'_y,u_t)\wedge$$
$$\psi_{(\mathcal{A}_{st},\mathcal{A}_t)}(u_{t_O},u_{t_E},u_0,u_1,u_2,v_{a_{1,1}},v_{a_{1,2}},v_{a_{2,1}},v_{a_{2,2}},v_{b_1},v_{b_2},u_t,v_\alpha,v_\beta,$$
$$v'_x,v'_y,u_t,v_x,v_y,v_t)))).$$

In the above formula, we omitted, for all point variables except v, the sub formulas expressing collinearity with v_0 and v_1. Also, the predicate **comp-coord** is an abbreviation for the fact that the translation of v's coordinates to the computation plane are v'_x, v'_y and u_t. The exact formula expressing this can be found in the proof of Lemma 4.24, when the translation of relation predicates is explained.

(iii) For the set $Type_{(\mathcal{A}_{st},\mathcal{A}_t)}(R^{st})=\{\alpha\in(\mathcal{A}_{st},\mathcal{A}_t)\mid\alpha(R^{st})=\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(R^{st})\}$, we compute two separate relations storing the $\mathcal{A}_t$-type, respectively $\mathcal{A}_{st}$-type of the encoded relation. In the previous formula, it was checked whether there exists a transformation mapping all points in R^{st} to points in the formula coded by m (*i.e.*, in F). Here, we compute that transformation:

$$T_{\mathcal{A}_t}:=\{(u_{t_O},u_{t_E},u_0,u_1,u_2,v_\alpha,v_\beta)\mid(\mathbf{TCoSys}_{\mathcal{A}}(u_{t_O},u_{t_E})\wedge$$
$$\mathbf{CoSys}^n_{\mathcal{A}}(u_0,u_1,u_2))\rightarrow\exists w\,(N(u_{t_O},u_{t_E},u_0,u_1,u_2,w)\wedge$$
$$\forall v_x\,\forall v_y\,\forall v_t\,(F(u_{t_O},u_{t_E},u_0,u_1,u_2,w,v_x,v_y,v_t)\leftrightarrow$$
$$\exists v\,\exists v'_x\,\exists v'_y\,\exists v'_t\,\exists v''_x\,\exists v''_y\,\exists v_{\alpha_{0,0}}\,\exists v_{\alpha_{0,1}}\,\exists v_{\alpha_{1,0}}\,\exists v_{\alpha_{1,1}}\,\exists v_{\beta_0}\,\exists v_{\beta_1})$$
$$(R^{st}(v)\wedge\mathbf{comp\text{-}coord}(u_{t_O},u_{t_E},u_0,u_1,u_2,v,v'_x,v'_y,v'_t)\wedge$$
$$\psi_{\mathcal{A}_{st}}(u_{t_O},u_{t_E},u_0,u_1,u_2,v_{\alpha_{0,0}},v_{\alpha_{0,1}},v_{\alpha_{1,0}},v_{\alpha_{1,1}},v_{\beta_0},v_{\beta_1},$$
$$v'_t,v'_x,v'_y,v'_t,v''_x,v''_y,v'_t)\wedge$$
$$\psi_{\mathcal{A}_t}(u_{t_O},u_{t_E},u_0,u_1,u_2,v_\alpha,v_\beta,v''_x,v''_y,v'_t,v_x,v_y,v_t))))\}$$

and

$$
\begin{aligned}
T_{\mathcal{A}_{st}} := \{(u_{t_O}, u_{t_E}, u_0, u_1, u_2, v_{\alpha_{0,0}}, v_{\alpha_{0,0}}, v_{\alpha_{0,1}}, v_{\alpha_{1,0}}, v_{\alpha_{1,1}}, v_{\beta_0}, v_{\beta_1}, u_t) \mid \\
(\mathbf{TCoSys}_{\mathcal{A}}(u_{t_O}, u_{t_E}) \wedge \mathbf{CoSys}^n_{\mathcal{A}}(u_0, u_1, u_2)) \rightarrow \\
\exists w\,(N(u_{t_O}, u_{t_E}, u_0, u_1, u_2, w) \wedge \forall v_x\,\forall v_y\,\forall v_t\,(\\
F(u_{t_O}, u_{t_E}, u_0, u_1, u_2, w, v_x, v_y, v_t) \leftrightarrow \exists v\,\exists v'_x\,\exists v'_y\,\exists v'_t\,\exists v_\alpha\,\exists v_\beta \\
(R^{st}(v) \wedge \mathbf{comp\text{-}coord}(u_{t_O}, u_{t_E}, u_0, u_1, u_2, v, v'_x, v'_y, v'_t) \wedge \\
T_{\mathcal{A}_t}(u_{t_O}, u_{t_E}, u_0, u_1, u_2, v_\alpha, v_\beta) \wedge \\
\psi_{\mathcal{A}_t}(u_{t_O}, u_{t_E}, u_0, u_1, u_2, v_\alpha, v_\beta, v'_x, v'_y, v'_t, v'_x, v'_y, v_t) \wedge \\
\psi_{\mathcal{A}_{st}}(u_{t_O}, u_{t_E}, u_0, u_1, u_2, v_{\alpha_{0,0}}, v_{\alpha_{0,1}}, v_{\alpha_{1,0}}, v_{\alpha_{1,1}}, v_{\beta_0}, v_{\beta_1}, \\
v_t, v'_x, v'_y, v_t, v_x, v_y, v_t))))\}.
\end{aligned}
$$

The decoding algorithm can be expressed in FO(Π, σ^{st}) + While.

Input databases are encoded by natural numbers. A counter machine simulates the query on this natural number and returns a natural number that encodes the output. In the decoding algorithm, again all natural numbers are enumerated and the evaluation of the terms and formulas they encode are stored in relations called T and F. When the number that is the output of the counter machine is encountered, the relation F contains all points of the result, up to the transformation stored in $\text{Type}_{(\mathcal{A}_{st},\mathcal{A}_t)}$ (because the query is $(\mathcal{A}_{st}, \mathcal{A}_t)-$generic). The result corresponds to the set $Q(\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F}))$. As Q is assumed to be a $(\mathcal{A}_{st}, \mathcal{A}_t)-$generic query, we have that for all $f \in Type_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F})$

$$Q(\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F})) = Q(f(\mathfrak{F})) = f(Q(\mathfrak{F})),$$

so $Q(\mathfrak{F})$ is computed as

$$\bigcup_{f \in Type_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F})} f^{-1}(Q(\mathbf{Canon}_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F}))) = \bigcup_{f \in Type_{(\mathcal{A}_{st},\mathcal{A}_t)}(\mathfrak{F})} f^{-1}(f(Q(\mathfrak{F}))).$$

For completeness, we give a program Decode that, when applied to the encoding N_φ of a formula φ, computes in a relation variable Dec the spatio-temporal relation defined by φ. Thereto it suffices to modify the encode program as shown in Algorithm 2.

The formula constructing the output, using the above, only differs slightly from the formulas we gave when explaining the encoding algorithm. In the encoding phase, it had to be checked, for some natural number m, whether there existed a transformation mapping all points of R^{st} to the points satisfying the formula encoded by m. Also, that transformation was computed. Here, we have the transformation stored in $\text{Type}_{(\mathcal{A}_{st},\mathcal{A}_t)}$, and we know we have the right natural number m, so all points mapped by the transformation in $\text{Type}_{(\mathcal{A}_{st},\mathcal{A}_t)}$ to points satisfying the formula encoded by m, are returned.

To conclude we summarize the conversion procedure. Given a k-ary computable query Q over a schema $\sigma^{st} = \{R^{st}_1, \ldots, R^{st}_m\}$, there exists a counter program M such

Algorithm 2 The decoding program. The input is a natural number encoding a relation.

$m := 0$
$T := \emptyset$
$F := \emptyset$
found :=False
while not found **do**
  $m := m + 1$
  build relations T and F
  found:= $m = N_\varphi$
end while
Dec := all points which are the image under the transformation stored in $\text{Type}_{(\mathcal{A}_{st},\mathcal{A}_t)}$ of the points with coordinates (represented as points on the line u_0u_1) p_x, p_y, p_t such that $F(u_{t_O}, u_{t_E}, u_0, u_1, u_2, m, p_1, p_2, p_t)$.

that for each database $\mathfrak{F}$ over σ^{st}, if $(n_{R_1^{st}}, \ldots, n_{R_m^{st}})$ are the results of applying the program Encode to $\mathfrak{F}$ then $M(n_{R_1^{st}}, \ldots, n_{R_m^{st}})$ is the encoding of the quantifier-free formula defining $Q(\mathfrak{F})$, using the variables $x_1^1, \ldots, x_1^{n+1}, \ldots, x_K^1, \ldots, x_K^{n+1}$. If $Q(\mathfrak{F})$ is not defined, then M does not halt on this input. As already noted above, we can simulate M by a program P in FO(Π, σ^{st}) + While. Hence, the query Q is expressed by the program

Encode;
P;
Decode;

The reason that the problem of identifying sound and complete languages for the groups $\mathcal{F}_{st}^f$ is still open, is that for those groups, there is no first-order logic formula expressing their graph. Indeed, it is not possible to express that there should exist a finite number of time moments for which there is a different affinity, when describing the groups $\mathcal{F}_{st}^f$. Hence, we cannot use the above proof technique.

The previous theorem has a number of corollaries.

Theorem 4.40. Let σ^{st} be a database schema. Let $(\mathcal{F}_{st}, \mathcal{F}_t)$ be one of the groups $(\mathcal{A}_{st}, \mathcal{A}_t)$, $(\mathcal{A}_{st}, \mathcal{I}_t)$, $(\mathcal{A}_{st}, Id_t)$, $(\mathcal{S}_{st}, \mathcal{F}_t)$, $(\mathcal{I}_{st}, \mathcal{F}_t)$, or $(\mathcal{T}_{st}, \mathcal{F}_t)$ with $\mathcal{F}_t \in \{\mathcal{A}_t, \mathcal{T}_t, Id_t\}$ and let $\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$ be as in Table 4.3. The point language FO($\Pi(\mathcal{F}_{st}, \mathcal{F}_t)$, σ)+While is sound and complete for the $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic computable spatio-temporal queries over σ.

Proof. The proof of this corollary is similar to the proof of Theorem 4.39. The encoding and decoding programs for the various transformation groups only differ where the transformation in $\text{Type}_{(\mathcal{A}_{st},\mathcal{A}_t)}$ is described, and where a coordinate system needs to be defined. The rest of the proof is the same, regardless of the transformation groups considered. The descriptions of the coordinate systems for the various transformation groups can be found in the proof of Corollary 4.26 □

5

The Parametric Spatio-temporal Data Model

In Chapter 3, we defined spatio-temporal data as a special sort of constraint data, as semi-algebraic sets in $(\mathbb{R}^n \times \mathbb{R})$. Although this model has nice properties, the query languages that were identified in Chapter 4 are not really user-friendly.

In this chapter, we define a more concrete data model for spatio-temporal data. Hereto, the new concept of spatio-temporal object is introduced. We represent a spatio-temporal object as a finite number of objects represented by means of a spatial reference object, a temporal object (*i.e.*, a time interval) and a time-dependent geometric transformation that determines how this spatial object moves or changes through space during the considered time interval. Although this model is suited for data in arbitrary dimensions, we focus on two-dimensional reference objects that move or change during time.

In this framework, a number of classes of practically relevant spatio-temporal objects arise naturally. These classes are indexed by the type of spatial reference object and the type of transformation functions that are allowed. On the level of reference objects, we consider polygons, triangles, triangles with two sides parallel to the coordinate axes of the two-dimensional plane and rectangles with all sides parallel to the coordinate axes. We consider time-dependent affinities, scalings and translations for what concerns transformation functions. These functions can be expressed by coefficients that are rational, polynomial, respectively linear functions of time.

We investigate these classes with respect to closure under Boolean set operations, namely union, intersection and set-difference.

By definition, these classes are closed under union (a spatio-temporal object is described as the union of atomic objects). We call a class closed under intersection (respectively set-difference) if any finite intersection (respectively set-difference) of

objects from a class can again be described by an object from that class (*i.e.*, as a union of atomic objects). The classes that we consider are not necessarily closed under intersection and set-difference.

We provide an in-depth and exhaustive study of their closure with respect to all set-theoretic operations, and we conclude that our model for representing spatio-temporal data gives very poor closure results for the classes of objects we considered important for spatio-temporal practice. The only classes that seems to be useful in this respect have polygons as spatial reference objects and use rational affinities to move or change these objects in time.

A conclusion is that we have to enrich the data model by allowing set-theoretic operations other than union in the construction of geometric objects from atomic geometric objects. As soon as we also allow spatio-temporal objects to be constructed from atomic ones by means of union and intersection (or union and set-difference) then the model becomes closed for *all* Boolean set operations. Indeed, as an important result, we show that our classes, that are drawn from practice, have the nice property that they are closed under intersection if and only if they are closed under set-difference.

To appreciate the need for applying set-theoretic operators to spatiotemporal objects, consider the following scenario. Let two spatial objects represent the extents of the safe areas around two different ships. Taking into account the movement of ships, the extents of the safe areas over a period of time can be represented as two spatiotemporal objects. To avoid collisions, one needs to be able to determine the intersection of those objects.

This chapter is organized as follows: In Section 2, we give definitions and describe the relevant classes of spatio-temporal objects. The closure results for these classes with respect to Boolean set operations are given in Section 3. We propose the extended model in Section 4 and describe a normal form for objects in this extended model. Section 5 gives comments and concludes the chapter.

5.1 Definitions

In this section, we define the notion of *spatio-temporal object*. In our approach, a spatio-temporal object consists of a spatial reference object, a time interval during which the spatio-temporal object exists and a continuous transformation that defines how the spatial reference object moves and changes during the interval of time.

5.1.1 Spatio-temporal and Geometric Objects

Definition 5.1 (**Spatio-temporal object**)**.** A *spatial object* is a subset of $\mathbb{R}^2$. A *temporal object* is a subset of $\mathbb{R}$ (we assume a single temporal dimension). A *spatio-temporal object* is a subset of $(\mathbb{R}^2 \times \mathbb{R})$.

These definitions are very general and disregard the fact that objects should be finitely representable in the computer's memory. In this chapter, we will study more restricted classes of spatial and spatio-temporal objects that are important from a

practical point of view and have simple and efficient representations. Such classes have been identified in the course of spatial and spatio-temporal database research.

Here, we propose a geometric approach: a spatio-temporal object is defined as a spatial reference object together with a continuous transformation that defines how the object moves or changes during some time interval.

Definition 5.2 (**Atomic geometric object**). An *atomic geometric object* $\mathcal{O}$ is a triple (S, I, f), where

- $S \subset \mathbb{R}^2$ is the *spatial reference object* of $\mathcal{O}$, which is semi-algebraic in $\mathbb{R}^2$;
- $I \subset \mathbb{R}$ is the *time domain* of $\mathcal{O}$, which is a connected and bounded semi-algebraic set in $\mathbb{R}$ (*i.e.*, a point or a bounded interval); and
- $f : (\mathbb{R}^2 \times \mathbb{R}) \to \mathbb{R}^2$ is the *transformation function* of $\mathcal{O}$, which is semi-algebraic and continuous both in the time coordinate and in the spatial coordinates.

The semantics of an atomic geometric object $\mathcal{O} = (S, I, f)$ is the spatio-temporal object $st(\mathcal{O}) =$

$$\{(x, y, t) \in (\mathbb{R}^2 \times \mathbb{R}) \mid \exists x' \, \exists y' \, ((x', y') \in S \wedge t \in I \wedge (x, y) = f(x', y', t))\}.$$

We remark that this definition guarantees that there is a finite representation of an atomic geometric object by means of the polynomial inequalities that describe its reference object, its time domain and the graph of its transformation function. This means that this data model is within the constraint model for databases.

Definition 5.3 (**Geometric object**). A *geometric object* is a finite set of atomic geometric objects. The semantics of a geometric object $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ is the union of the semantics of the atomic objects that constitute it, *i.e.*, the set

$$\bigcup_{1 \le i \le n} st(\mathcal{O}_i).$$

We agree that whenever we write "the spatio-temporal object $\mathcal{O}$", where $\mathcal{O}$ is an (atomic) geometric object, we mean the semantics of the (atomic) geometric object $\mathcal{O}$. Also, when $\mathcal{O} = (S, I, f)$ is an atomic object and $\tau \in I$, we will refer to the set $\{(x, y) \mid \exists x' \, \exists y' \, ((x', y') \in S \wedge (x, y) = f(x', y', \tau))\}$ as the *snapshot of* $\mathcal{O}$ *at time* τ and we will denote it $f(S, \tau)$ or $\mathcal{O}^\tau$ (see also Definition 3.14).

We define the *time domain* of a geometric object to be the smallest time interval that contains all the time domains of the composing atomic geometric objects. Recall that the smallest interval containing a set of intervals is also known as the convex closure of this set. We denote the convex closure of the sets I_1, I_2, ..., I_n by $\overline{\bigcup}_{i=1}^{n} I_i$.

Remark that a spatio-temporal object is empty (or non-existing) outside the time domain of the geometric object that defines it. Also, within its time domain a spatio-temporal object can be empty (for instance, at any moment when no atomic geometric object exists).

We conclude this section by remarking that the above introduced notions of spatial and spatio-temporal object and of (atomic) geometric object can be generalized to arbitrary dimension d (by simply substituting d for 2 in the above definitions). Since all the results in this chapter are formulated for dimension 2, we have chosen not to use this generalization here.

5.1.2 Practically Relevant Classes of Geometric Objects

Here, we define special classes of geometric objects that are relevant to spatio-temporal database practice. These classes are denoted by

$$\langle \mathcal{S}, \mathcal{F} \rangle$$

and they are determined by the type $\mathcal{S}$ of spatial reference object and the type $\mathcal{F}$ of transformation function. For clarity, a geometric object belongs to a class if all of its atomic geometric objects belong to that class.

The classes of geometric figures in the plane $\mathbb{R}^2$ that we will consider are

- $\mathcal{S}_{\mathrm{Poly}}$, the class of arbitrary polygons,
- $\mathcal{S}_{\mathrm{Tr}}$, the class of arbitrary triangles,
- $\mathcal{S}_{\mathrm{TrAx}}$, the class of triangles with two sides parallel to the coordinate axes, and
- $\mathcal{S}_{\mathrm{Rect}}$ the class of rectangles with all sides parallel to the coordinate axes.

In this thesis, we assume triangles, polygons and rectangles to be filled objects. But since we allow two or more corner points of a triangle or rectangle to coincide, the model can deal with polylines and points too. A line segment and a point are considered triangles. Also line segments parallel to the axes and points are considered rectangles. Finally, note that $\mathcal{S}_{\mathrm{Rect}} \subset \mathcal{S}_{\mathrm{TrAx}} \subset \mathcal{S}_{\mathrm{Tr}} \subset \mathcal{S}_{\mathrm{Poly}}$.

The classes of transformation functions we will consider are

- $\mathcal{F}_{\mathrm{Aff}}$, the class of the affine transformations,
- $\mathcal{F}_{\mathrm{Sc}}$, the class of the scalings,
- $\mathcal{F}_{\mathrm{Trans}}$, the class of the translations, and
- $\mathcal{F}_{\mathrm{id}}$, the class consisting of the identity mapping.

It is clear that $\mathcal{F}_{\mathrm{id}}$, $\mathcal{F}_{\mathrm{Trans}}$ and $\mathcal{F}_{\mathrm{Sc}}$ are subclasses of $\mathcal{F}_{\mathrm{Aff}}$. More technically, these classes are defined as follows. The class $\mathcal{F}_{\mathrm{Aff}}$ of affine transformations consists of the mappings $(\mathbb{R}^2 \times \mathbb{R}) \to \mathbb{R}^2$ of the form

$$(x, y, t) \mapsto \begin{pmatrix} a(t) & b(t) \\ c(t) & d(t) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e(t) \\ f(t) \end{pmatrix},$$

where a, b, c, d, e, and f are function from $\mathbb{R}$ to $\mathbb{R}$ with $a(t)d(t) - c(t)b(t) \neq 0$ for all t in the relevant time domain.

The class $\mathcal{F}_{\mathrm{Sc}}$ of scalings consists of the affine transformations for which the functions b and c are identical to 0. The class $\mathcal{F}_{\mathrm{Trans}}$ consists of the scalings for which the functions a and d are identical to 1.

For practical purposes we will only consider functions a, b, c, d, e, and f that are semi-algebraic and continuous as required by the definition. These are

- the rational functions (*i.e.*, fractions of polynomial functions),

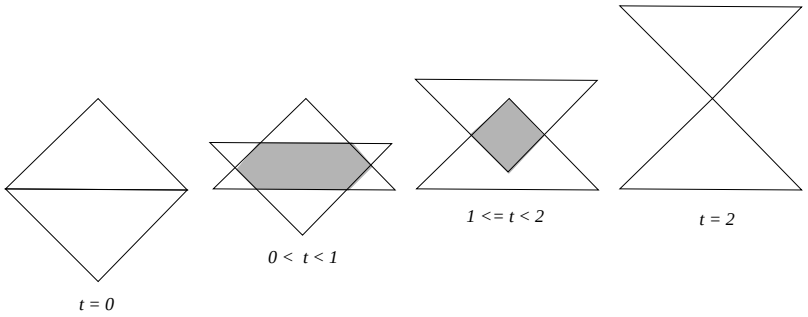


Figure 5.1: Two atomic geometric objects. The time domain can be partitioned in four parts such that the intersection of the two objects retains the same shape during each element of the partition.

- the polynomial functions and
- the linear polynomial functions.

The corresponding classes of transformations will be denoted using superscripts $\mathcal{F}^{\mathrm{Rat}}$, $\mathcal{F}^{\mathrm{Poly}}$, and $\mathcal{F}^{\mathrm{Lin}}$. For example, $\mathcal{F}^{\mathrm{Rat}}_{\mathrm{Sc}}$ represents the class of rational scalings. We assume that the time domain of an atomic geometric object belongs to the domain of the transformation function and that the denominator of a rational function in the definition of a transformation is never zero in the closure of the time domain (thus, the moving figure will remain within fixed bounds during the time domain).

Note that the shape of a spatio-temporal object at a certain time instant is not necessarily the same as the shape of the reference object of the geometric object that gives rise to the spatio-temporal object. For example, a rectangle is mapped to a parallelogram under an affinity.

5.1.3 Example

Let $\mathcal{O}_A = (S_A, I_A, f_A)$ and $\mathcal{O}_B = (S_B, I_B, f_B)$ be two (atomic) geometric objects with spatial reference objects S_A and S_B respectively the triangles with corner points $(-1, 0)$, $(1, 0)$, $(0, 1)$ and $(-1, 0)$, $(1, 0)$, $(0, -1)$, and time domains $I_A = I_B = [0, 2]$. In this time domain, S_A remains at its place (*i.e.*, $f_A(x, y, t) = (x, y)$ for all t), while S_B is translated with constant speed (equal to 1) in the direction of the positive y-axis (*i.e.*, $f_B(x, y, t) = (x, y + t)$. The functions f_A and f_B belong to $\mathcal{F}^{\mathrm{Lin}}_{\mathrm{Trans}}$.

At $t = 0$ both objects intersect in a line segment. For $0 < t < 1$ they intersect in a hexagon, for $1 \leq t < 2$ in a quadrangle, and finally for $t = 2$ in a point. Figure 5.1 illustrates their movement.

5.2 Closure Properties under Boolean Set Operations

In this section, we work with the classes $\langle \mathcal{S}, \mathcal{F} \rangle$ introduced in the previous section, and we investigate which of these classes $\langle \mathcal{S}, \mathcal{F} \rangle$ are closed under the Boolean set operations $\cup$ (union), $\cap$ (intersection) and $\setminus$ (set difference). We first define what closure means.

Definition 5.4 (Closure). Let θ be one of the operations $\cup$, $\cap$ or $\setminus$. We say that the class $\langle \mathcal{S}, \mathcal{F} \rangle$ is *(atomically) closed under* θ if for any two (atomic) geometric objects $\mathcal{O}_1$ and $\mathcal{O}_2$ in $\langle \mathcal{S}, \mathcal{F} \rangle$ there exists a geometric object $\mathcal{O}$ in $\langle \mathcal{S}, \mathcal{F} \rangle$ such that $st(\mathcal{O}) = st(\mathcal{O}_1)\ \theta\ st(\mathcal{O}_2)$.

We will refer to an object $\mathcal{O}$ that satisfies the condition in the definition as an intersection, union or difference of $\mathcal{O}_1$ and $\mathcal{O}_2$ (they need not be unique).

For the union operation, the closure follows immediately from the definition of geometric objects (Definition 5.3).

Property 5.2.1 (Closure for $\cup$). For any class of objects $\mathcal{S}$ and any class of transformations $\mathcal{F}$, $\langle \mathcal{S}, \mathcal{F} \rangle$ is closed under $\cup$.

For $\cap$ and $\setminus$ the situation is more complicated. The next theorem is the main result that we want to prove in this section. It summarizes the closure results for intersection and set-difference.

Theorem 5.5 (Closure for $\cap$ and $\setminus$). For any class of objects $\mathcal{S}$ among $\mathcal{S}_{\text{Poly}}$, $\mathcal{S}_{\text{Tr}}$, $\mathcal{S}_{\text{TrAx}}$ and $\mathcal{S}_{\text{Rect}}$ and any class of transformations $\mathcal{F}$ among $\mathcal{F}_{\text{Aff}}$, $\mathcal{F}_{\text{Sc}}$, $\mathcal{F}_{\text{Trans}}$ and $\mathcal{F}_{\text{id}}$, the closure with respect to $\cap$ and $\setminus$ is summarized in the following table.

$\cap, \setminus$	$\mathcal{F}_{\text{Aff}}^{\text{Rat}}$	$\mathcal{F}_{\text{Aff}}^{\text{Pol}}$	$\mathcal{F}_{\text{Aff}}^{\text{Lin}}$	$\mathcal{F}_{\text{Sc}}^{\text{Rat}}$	$\mathcal{F}_{\text{Sc}}^{\text{Pol}}$	$\mathcal{F}_{\text{Sc}}^{\text{Lin}}$	$\mathcal{F}_{\text{Trans}}^{\text{Rat}}$	$\mathcal{F}_{\text{Trans}}^{\text{Pol}}$	$\mathcal{F}_{\text{Trans}}^{\text{Lin}}$	$\mathcal{F}_{\text{id}}$
$\mathcal{S}_{\text{Poly}}$	+	−	−	−	−	−	−	−	−	+†
$\mathcal{S}_{\text{Tr}}$	+	−	−	−	−	−	−	−	−	+†
$\mathcal{S}_{\text{TrAx}}$	+	−	−	−	−	−	−	−	−	−
$\mathcal{S}_{\text{Rect}}$	+	−	−	+	+	+	−	−	−	+

Closure is indicated by a + sign, non-closure by a − sign.

The items marked with † are from [74]. The remainder of this section is devoted to proving this theorem. We do this by first proving some lemmas in a first subsection that reduce the number of cases that have to be looked at and by then proving the remaining cases in a second subsection.

5.2.1 Reduction Properties

The properties in this section reduce the number of cases that have to be investigated. First, we give a set-theoretic lemma that will be used frequently.

Lemma 5.6 (Set-theoretical observations). Let $A_1, \ldots, A_n$ and $B_1, \ldots, B_m$ be sets. Then

(i) $(\bigcup_{i=1}^{n} A_i) \cap (\bigcup_{j=1}^{m} B_j) = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} (A_i \cap B_j)$,

(ii) $(\bigcup_{i=1}^{n} A_i) \setminus (\bigcup_{j=1}^{m} B_j) = \bigcup_{i=1}^{n} ((\cdots((A_i \setminus B_1) \setminus B_2) \setminus \cdots) \setminus B_m)$

Proof. The first equality follows directly from distributivity of intersection with respect to union.

The second equality can be proven by induction on m, using the observation that

$$(\bigcup_{i=1}^{n} A_i) \setminus B = (\bigcup_{i=1}^{n} A_i) \cap B^c = \bigcup_{i=1}^{n} (A_i \cap B^c) = \bigcup_{i=1}^{n} (A_i \setminus B),$$

where B^c denotes the complement of B with respect to some universe. □

The next property says that for $\cap$ and $\setminus$ closure and closure on atomic objects coincide.

Property 5.2.2 (Atomicity). Let $\mathcal{S}$ be a class of objects and $\mathcal{F}$ a class of transformations. Then

(i) $\langle \mathcal{S}, \mathcal{F} \rangle$ is closed under $\cap$ if and only if it is atomically closed under $\cap$, and

(ii) $\langle \mathcal{S}, \mathcal{F} \rangle$ is closed under $\setminus$ if and only if it is atomically closed under $\setminus$.

Proof. Both for (i) and (ii) the only-if direction is obvious. So we concentrate on the if-direction.

For the if-direction of (i), assume that $\langle \mathcal{S}, \mathcal{F} \rangle$ is atomically closed under $\cap$ and let $\{\mathcal{O}_{1,1}, \mathcal{O}_{1,2}, \ldots, \mathcal{O}_{1,n}\}$ and $\{\mathcal{O}_{2,1}, \mathcal{O}_{2,2}, \ldots, \mathcal{O}_{2,m}\}$ be two geometric objects from $\langle \mathcal{S}, \mathcal{F} \rangle$. By using Lemma 5.6 (i), we get

$$(\bigcup_{i=1}^{n} st(\mathcal{O}_{1,i})) \cap (\bigcup_{j=1}^{m} st(\mathcal{O}_{2,j})) = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m} (st(\mathcal{O}_{1,i}) \cap st(\mathcal{O}_{2,j})).$$

Since $\cap$ is assumed to be atomically closed, each $st(\mathcal{O}_{1,i}) \cap st(\mathcal{O}_{2,j})$ can be written as a union $\bigcup_{k=1}^{l_{ij}} st(\mathcal{O}_{k,i,j})$, where each $\mathcal{O}_{k,i,j}$ is an atomic geometric object. Therefore, the intersection of $\{\mathcal{O}_{1,1}, \mathcal{O}_{1,2}, \ldots, \mathcal{O}_{1,n}\}$ and $\{\mathcal{O}_{2,1}, \mathcal{O}_{2,2}, \ldots, \mathcal{O}_{2,m}\}$ can also be written as $\bigcup_{i=1}^{n} \bigcup_{j=1}^{m} \bigcup_{k=1}^{l_{i,j}} st(\mathcal{O}_{k,i,j})$. This completes the proof of the if-direction of (i).

For the if-direction of (ii), assume that $\langle \mathcal{S}, \mathcal{F} \rangle$ is atomically closed under $\setminus$ and let $\{\mathcal{O}_{1,1}, \mathcal{O}_{1,2}, \ldots, \mathcal{O}_{1,n}\}$ and $\{\mathcal{O}_{2,1}, \mathcal{O}_{2,2}, \ldots, \mathcal{O}_{2,m}\}$ be two geometric objects from $\langle \mathcal{S}, \mathcal{F} \rangle$. By using Lemma 5.6 (ii), we get

$$(\bigcup_{i=1}^{n} st(\mathcal{O}_{1,i})) \setminus (\bigcup_{j=1}^{m} st(\mathcal{O}_{2,j})) = \bigcup_{i=1}^{n} ((\cdots((st(\mathcal{O}_{1,i}) \setminus st(\mathcal{O}_{2,1})) \setminus st(\mathcal{O}_{2,2})) \setminus \cdots) \setminus st(\mathcal{O}_{2,m})).$$

We prove, by induction on m, that $((\cdots((st(\mathcal{O}_{1,i}) \setminus st(\mathcal{O}_{2,1})) \setminus st(\mathcal{O}_{2,2})) \setminus \cdots) \setminus st(\mathcal{O}_{2,m}))$ is of the form $\bigcup_{k=1}^{l} st(\mathcal{O}'_k)$. Since $\setminus$ is assumed to be atomically closed, $st(\mathcal{O}_{1i}) \setminus st(\mathcal{O}_{21})$ can be written as a union $\bigcup_{k=1}^{l_1} st(\mathcal{O}'_k)$, where each $\mathcal{O}'_k$ is an atomic geometric object. This proves the case $m = 1$. Next, assume we have shown that $((\cdots((st(\mathcal{O}_{1,i}) \setminus st(\mathcal{O}_{2,1})) \setminus st(\mathcal{O}_{2,2})) \setminus \cdots) \setminus st(\mathcal{O}_{2,m-1}))$ is $\bigcup_{k=1}^{l} st(\mathcal{O}'_k)$ with all $\mathcal{O}'_k$

atomic geometric objects. Then $((\cdots((st(\mathcal{O}_{1,i}) \setminus st(\mathcal{O}_{2,1})) \setminus st(\mathcal{O}_{2,2})) \setminus \cdots) \setminus st(\mathcal{O}_{2,m}))$ is $(\bigcup_{k=1}^{l} st(\mathcal{O}'_k)) \setminus st(\mathcal{O}_{2,m})$, which is $\bigcup_{k=1}^{l}(st(\mathcal{O}'_k) \setminus st(\mathcal{O}_{2,m}))$, using Lemma 5.6 (ii). Again, since $\setminus$ is assumed to be atomically closed, each of the sets $st(\mathcal{O}'_k) \setminus st(\mathcal{O}_{2,m})$ is of the form $\bigcup_{r=1}^{l_k} st(\mathcal{O}''_r)$. Therefore, the set-difference of $\{\mathcal{O}_{1,1}, \mathcal{O}_{1,2}, \ldots, \mathcal{O}_{1,n}\}$ and $\{\mathcal{O}_{2,1}, \mathcal{O}_{2,2}, \ldots, \mathcal{O}_{2,m}\}$ is also the semantics of a geometric object from $\langle \mathcal{S}, \mathcal{F} \rangle$. This completes the proof. □

The following property states that intersection and set-difference are equivalent with respect to closure.

Property 5.2.3 (Equivalence of $\cap$ and $\setminus$). Let $\mathcal{S}$ be a class of objects and $\mathcal{F}$ a class of transformations. Then the class $\langle \mathcal{S}, \mathcal{F} \rangle$ is closed under $\cap$ if and only if it is closed under $\setminus$.

Proof. By the atomicity property (Property 5.2.2) it suffices to prove this property for atomic geometric objects.

For the if-direction, assume that $\langle \mathcal{S}, \mathcal{F} \rangle$ is closed under $\setminus$ and let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two atomic geometric objects from $\langle \mathcal{S}, \mathcal{F} \rangle$. Since,

$$st(\mathcal{O}_1) \cap st(\mathcal{O}_2) = (st(\mathcal{O}_1) \cup st(\mathcal{O}_2)) \setminus ((st(\mathcal{O}_1) \setminus st(\mathcal{O}_2)) \cup (st(\mathcal{O}_2) \setminus st(\mathcal{O}_1)))$$

and since $(st(\mathcal{O}_1) \setminus st(\mathcal{O}_2))$ and $(st(\mathcal{O}_2) \setminus st(\mathcal{O}_1))$ are by assumption $\bigcup_{i=1}^{n} st(\mathcal{O}'_i)$ respectively $\bigcup_{j=1}^{n} st(\mathcal{O}''_j)$ with all $\mathcal{O}'_i$ and $\mathcal{O}''_j$ atomic geometric objects. Therefore, $st(\mathcal{O}_1) \cap st(\mathcal{O}_2)$ equals $((\cdots(st(\mathcal{O}_2) \setminus st(\mathcal{O}'_1)) \setminus \cdots) \setminus st(\mathcal{O}'_n)) \cup ((\cdots(st(\mathcal{O}_2) \setminus st(\mathcal{O}''_1)) \setminus \cdots) \setminus st(\mathcal{O}''_m))$, using Lemma 5.6 (ii). Using the argumentation from the proof of the if-direction of (ii) of Property 5.2.2, we can show that this set is again a union of semantics of atomic geometric objects from $\langle \mathcal{S}, \mathcal{F} \rangle$.

For the only-if direction, assume that $\langle \mathcal{S}, \mathcal{F} \rangle$ is closed under $\cap$ and let $\mathcal{O}_1 = (S_1, I_1, f_1)$ and $\mathcal{O}_2 = (S_2, I_2, f_2)$ be two atomic geometric objects from $\langle \mathcal{S}, \mathcal{F} \rangle$. We have to show that $st(\mathcal{O}_1) \setminus st(\mathcal{O}_2)$ can be written as $\bigcup_{i=1}^{n} st(\mathcal{O}'_i)$, with $\mathcal{O}'_i$ atomic geometric objects. We can restrict our attention to the set $st(\mathcal{O}_1) \setminus st(\mathcal{O}_2)$ in the interval $I_1 \cap I_2$ rather than in the complete interval $I_1 \overline{\cup} I_2$ (since the set-difference is empty in $I_2 \setminus I_1$ and equal to $\mathcal{O}_1$ in $I_1 \setminus I_2$). Let I denote the topological closure of $I_1 \cap I_2$. The set $S_B = \{(x, y) \in \mathbb{R}^2 \mid \exists x' \exists y' \exists t\, ((x', y') \in S_1 \wedge t \in I \wedge f_1(x', y', t) = f_2(x, y, t))\}$ is compact (*i.e.*, topologically closed and bounded) since it is the image of the compact set $S_1 \times I$ under the continuous function $f_2^{-1} \circ f_1$. Therefore, also $S = S_2 \cup S_B$ is a compact set in $\mathbb{R}^2$. Let $\alpha : (x, y) \mapsto (ax + b_1, ay + b_2)$ be a scaling followed by a translation that maps S_2 to a set that strictly contains S (this is possible since S is bounded). Remark that α maps any line to a parallel line. Let $\mathcal{O}_3$ be the atomic geometric object $(\alpha(S_2), I, f_2)$. At any moment t in I, we thus have that $f_2(S_2, t) \subset f_2(\alpha(S_2), t)$ (since affinities are monotone mappings) and $f_1(S_1, t) \subset f_2(\alpha(S_2), t)$. Therefore, $st(\mathcal{O}_1) \setminus st(\mathcal{O}_2) = st(\mathcal{O}_1) \cap (st(\mathcal{O}_3) \setminus st(\mathcal{O}_2))$.

Now, $st(\mathcal{O}_3) \setminus st(\mathcal{O}_2)$ can always be written as the semantics of a geometric object in $\langle \mathcal{S}, \mathcal{F} \rangle$ where $\mathcal{S}$ and $\mathcal{F}$ are any pairs allowed in Theorem 5.5. For each of the classes $\mathcal{S}_{\text{Poly}}$, $\mathcal{S}_{\text{Tr}}$, $\mathcal{S}_{\text{TrAx}}$, and $\mathcal{S}_{\text{Rect}}$ this is illustrated in Figure 5.2. For each of these classes $\alpha(S_2) \setminus S_2$ can be partitioned into a finite number of reference objects T_1, ..., T_n from these classes. So, define the atomic geometric objects $\mathcal{O}'_i = (T_i, I, f_2)$ $(1 \leq i \leq n)$. Then $st(\mathcal{O}_3) \setminus st(\mathcal{O}_2) = \bigcup_{i=1}^{n} st(\mathcal{O}'_i)$. Therefore, $st(\mathcal{O}_1) \cap (st(\mathcal{O}_3) \setminus st(\mathcal{O}_2)) = st(\mathcal{O}_1) \cap$

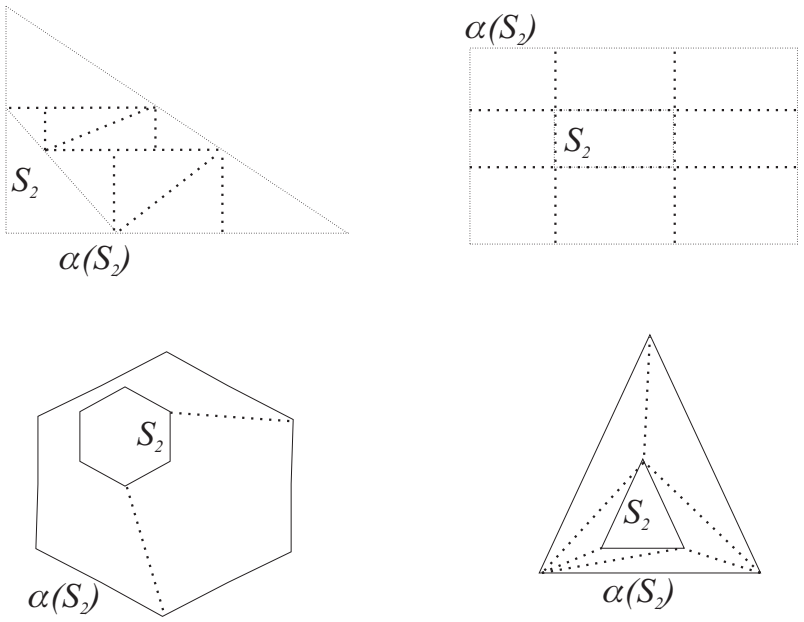


Figure 5.2: Examples of partitions of the set $\alpha(S_2) \setminus S_2$ for the classes $\mathcal{S}_{\text{Poly}}$, $\mathcal{S}_{\text{Tr}}$, $\mathcal{S}_{\text{TrAx}}$, and $\mathcal{S}_{\text{Rect}}$ respectively.

$\bigcup_{i=1}^{n} st(\mathcal{O}'_i) = \bigcup_{i=1}^{n}(st(\mathcal{O}_1) \cap st(\mathcal{O}'_i))$. Since we have assumed that $\langle \mathcal{S}, \mathcal{F} \rangle$ is closed for intersection, the intersections $st(\mathcal{O}_1) \cap st(\mathcal{O}'_i)$ can be written as $\bigcup_{k=1}^{l_i} st(\mathcal{O}''_k)$ with $\mathcal{O}''_k$ atomic geometric objects from $\langle \mathcal{S}, \mathcal{F} \rangle$. Therefore also $st(\mathcal{O}_1) \cap (st(\mathcal{O}_3) \setminus st(\mathcal{O}_2))$ can be written as such a union. This completes the proof. □

A final reduction property says that the closure results for polygons and triangles coincide. We can therefore concentrate on triangles further on.

Property 5.2.4 (Reduction of $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F} \rangle$ to $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F} \rangle$). Let $\mathcal{F}$ be a class of transformations, and let θ be one of the operations $\cup$, $\cap$ or $\setminus$. Then $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F} \rangle$ is closed under θ if and only if $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F} \rangle$ is closed under θ.

Proof. This property follows from the fact that any atomic geometric object $\mathcal{O} = (S, I, f)$ from $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F} \rangle$ corresponds to a geometric object from $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F} \rangle$. Indeed, let $T_1, \ldots, T_n$ be an arbitrary triangulation of the polygon S. The geometric object $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ with $\mathcal{O}_i = (T_i, I, f)$ $(1 \leq i \leq n)$ has the same semantics as $\mathcal{O} = (S, I, f)$.

So, if $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F} \rangle$ is closed under θ, then also any union, intersection or set-difference of two elements of $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F} \rangle$ is again a geometric object of $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F} \rangle$ and because of the above argument also of $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F} \rangle$.

On the other hand, suppose that $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F} \rangle$ is closed under θ. If $\mathcal{O}_1$ and $\mathcal{O}_2$ are objects in $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F} \rangle$, then so are their union, intersection or set-difference, since they are in $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F} \rangle$, which is a subclass of $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F} \rangle$. □

5.2.2 Closure and Non-closure Proofs

In this section, we complete the proof of Theorem 5.5, by means of a series of lemmas that cover all the cases presented in the matrix of Theorem 5.5. Here, we take the reduction results of the previous section into account. In particular, we only consider intersections *or* set-differences of *atomic* geometric objects, and we do not have to consider polygons any more.

5.2.2.1 Finite Time Partition

Before giving these lemmas we introduce the technical notion of *finite time partition*. This will be of use in many of the proofs in this section. The finite time partition property tells us how and when the form (or appearance) of the intersection or set-difference of two atomic geometric objects changes. We observe that the intersection of two moving triangles can be empty, a single point, a straight line segment, a triangle, a quadrangle, a pentagon and a hexagon. The intersection of two moving rectangles can be empty, a single point, a line segment or a rectangle. We refer to all these different forms of the intersection or the set-difference as their possible *shapes*. Also the difference of two triangles or two rectangles can take a finite number of different shapes. In the example in Figure 5.1, the intersection takes four different shapes, whereas the difference takes five different shapes.

We define this notion now more technically. Let $\mathcal{O}_1 = (S_1, I_1, f_1)$ and $\mathcal{O}_2 = (S_2, I_2, f_2)$ be two atomic geometric objects with rational affine transformations with time domains I_1 and I_2. In the following, we denote by $I_1 \,\overline{\cup}\, I_2$ the convex closure of the set $I_1 \cup I_2$ in $\mathbb{R}$. Let τ be in $I_1 \,\overline{\cup}\, I_2$. Firstly, we call the set of lines that intersect the border of $f_i(S_i, \tau)$ in infinitely many points, the set of *carriers of the frame* $f_i(S_i, \tau)$ and denote it $car(f_i(S_i, \tau))$ $(i = 1, 2)$.

Definition 5.7 (**Finite time partition**)**.** We call a *finite time partition of* $\mathcal{O}_1$ *and* $\mathcal{O}_2$ any partition of the interval $I_1 \overline{\cup} I_2$ into a finite number of time intervals $J_1, \ldots, J_m$ such that for any $\tau, \tau' \in J_i$ (and all $1 \leq i \leq m$), $car(f_1(S_1, \tau)) \cup car(f_2(S_2, \tau))$ and $car(f_1(S_1, \tau')) \cup car(f_2(S_2, \tau'))$ are topologically equivalent sets[1] in $\mathbb{R}^2$.

Property 5.2.5 (**Existence of finite time partition**)**.** Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be two atomic geometric objects with rational affine transformations with time domains I_1 and I_2. There exists a finite time partition of $\mathcal{O}_1$ and $\mathcal{O}_2$.

Proof. Let $\mathcal{O}_1 = (S_1, I_1, f_1)$ and $\mathcal{O}_2 = (S_2, I_2, f_2)$ be two atomic geometric objects satisfying the conditions of the statement of this property. From the assumption that the reference objects S_1 and S_2 are semi-algebraic and the transformation functions $f1$ and f_2 are affine rational functions, it follows that the sets $st(\mathcal{O}_1)$ and $st(\mathcal{O}_2)$ are semi-algebraic subsets of $(\mathbb{R}^2 \times \mathbb{R})$ (for details on this type of basic results on semi-algebraic sets, we refer to Chapter 2 of [8]). Let I be the set $I_1 \,\overline{\cup}\, I_2$.

Also, the set $A = \bigcup_{t \in I_1 \overline{\cup} I_2} (car(f_1(S_1, t)) \cup car(f_2(S_2, t)))$ is semi-algebraic, since it can be defined in the first-order logic of the reals over the semi-algebraic sets $st(\mathcal{O}_1)$ and $st(\mathcal{O}_2)$ (this closure property of first-order logic over the reals can be found in

[1] We call two subsets A and B of $\mathbb{R}^2$ *topologically equivalent* when there exists an orientation-preserving homeomorphism h of $\mathbb{R}^2$ such that $h(A) = B$.

Chapter 2 of [59]). We can therefore consider the set A as a subset of $(\mathbb{R}^2 \times \mathbb{R})$ parameterized by the time parameter t. It follows from *Semi-algebraic Triviality* (See Theorem 2.13) that the set A induces a finite partition on $I_1 \cup I_2$ such that in each partition class A remains topologically equivalent. □

5.2.2.2 Technical Lemmas

The following two lemmas are technical lemmas that say that two/three points that move with their respective rational affinities can be combined into one line/triangle that moves by a single rational affinity.

Lemma 5.8. Let $\mathcal{O}_i = (\{(x_i, y_i)\}, I, g_i)$ $(i = 1, 2, 3)$ be three atomic geometric objects with $g_i \in \mathcal{F}_{\text{Aff}}^{\text{Rat}}$. If the three points (x_1, y_1), (x_2, y_2) and (x_3, y_3) form a triangle S (*i.e.*, are not collinear) and if $g_1(x_1, y_1, t)$, $g_2(x_2, y_2, t)$ and $g_3(x_3, y_3, t)$ form a triangle S_t at any moment $t \in I$ (*i.e.*, are not collinear), then there exists an atomic geometric $\mathcal{O} = (S, I, g)$ with $g \in \mathcal{F}_{\text{Aff}}^{\text{Rat}}$ such that $g_i(x_i, y_i, t) = g(x_i, y_i, t)$ for all $t \in I$ and $i = 1, 2, 3$.

Proof. Let (x_1, y_1), (x_2, y_2) and (x_3, y_3) be the three corner points of the triangle S and let (x_i, y_i) be transformed by the affinity g_i given by

$$\begin{pmatrix} a_i(t) & b_i(t) \\ c_i(t) & d_i(t) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i(t) \\ f_i(t) \end{pmatrix}, \quad i = 1, 2, 3.$$

The condition for the existence of a single affine transformation that transforms these corner points according to their respective affinities is that the first matrix in the matrix equation below is regular.

$$\begin{pmatrix} x_1 & y_1 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_1 & y_1 & 0 & 1 \\ x_2 & y_2 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_2 & y_2 & 0 & 1 \\ x_3 & y_3 & 0 & 0 & 1 & 0 \\ 0 & 0 & x_3 & y_3 & 0 & 1 \end{pmatrix} \begin{pmatrix} a(t) \\ b(t) \\ c(t) \\ d(t) \\ e(t) \\ f(t) \end{pmatrix} = \begin{pmatrix} a_1(t)x_1 + b_1(t)y_1 + e_1(t) \\ c_1(t)x_1 + d_1(t)y_1 + f_1(t) \\ a_2(t)x_2 + b_2(t)y_2 + e_2(t) \\ c_2(t)x_2 + d_2(t)y_2 + f_2(t) \\ a_3(t)x_3 + b_3(t)y_3 + e_3(t) \\ c_3(t)x_3 + d_3(t)y_3 + f_3(t) \end{pmatrix}$$

This is the case if and only if the three points (x_1, y_1), (x_2, y_2) and (x_3, y_3) are not collinear. By assumption, this condition is satisfied. We find the affine transformation that transforms the triangle S according to the different movements of the corner points, by solving the above matrix equation.

The result of this computation is the affine transformation with coefficients $a(t)$, $b(t)$, $c(t)$, $d(t)$, $e(t)$, and $f(t)$ that have the following form (to save space time dependence is omitted):

First, we abbreviate the expression

$$x_1y_2 - x_1y_3 + x_2y_3 - x_3y_2 + x_3y_1 - x_2y_1$$

by D. The coefficients $a(t)$, $b(t)$, $c(t)$, $d(t)$, $e(t)$, and $f(t)$ are all of the form $\frac{a'(t)}{D}$, $\frac{b'(t)}{D}$, $\frac{c'(t)}{D}$, $\frac{d'(t)}{D}$, $\frac{e'(t)}{D}$, and $\frac{f'(t)}{D}$, respectively, where

$$a'(t) = -x_1a_1y_3 + x_1a_1y_2 - y_2b_3y_3 + e_2y_3 - e_1y_3 - y_2e_3 + e_1y_2 + y_1a_3x_3 + a_2x_2y_3 + y_1e_3 - y_2a_3x_3 + b_2y_2y_3 - y_1a_2x_2 + y_1b_3y_3 - y_1e_2 - y_1b_2y_2 + b_1y_1y_2 - b_1y_1y_3,$$

$$b'(t) = x_1b_2y_2 + x_2a_3x_3 - x_2b_1y_1 - x_1b_3y_3 + x_3a_1x_1 + x_1a_2x_2 + x_2b_3y_3 + x_2e_3 - x_3a_2x_2 + x_3e_1 - x_3b_2y_2 - x_1a_3x_3 - x_2a_1x_1 - x_1e_3 - x_2e_1 + x_3b_1y_1 - x_3e_2 + x_1e_2,$$

$$c'(t) = x_1c_1y_2 - x_1c_1y_3 - y_1c_2x_2 + y_1f_3 - y_2f_3 + f_1y_2 + y_1c_3x_3 - y_2d_3y_3 + y_1d_3y_3 - y_1d_2y_2 + d_1y_1y_2 + f_2y_3 - y_2c_3x_3 - y_1f_2 - d_1y_1y_3 + c_2x_2y_3 + d_2y_2y_3 - f_1y_3,$$

$$d'(t) = -x_2d_1y_1 - x_1d_3y_3 + x_3d_1y_1 - x_3d_2y_2 + x_1c_2x_2 - x_3c_2x_2 - x_2c_1x_1 + x_2f_3 + x_3f_1 - x_1c_3x_3 + x_2c_3x_3 + x_2d_3y_3 + x_1d_2y_2 + x_3c_1x_1 - x_2f_1 - x_1f_3 + x_1f_2 - x_3f_2,$$

$$\begin{aligned} e'(t) = {} & y_1x_3e_2 + y_2x_1e_3 - y_2x_3e_1 - y_2x_3a_1x_1 + y_2x_1b_3y_3 - e_2x_1y_3 \\ & + e_1x_2y_3 + b_1y_1x_2y_3 - a_2x_2x_1y_3 - y_1x_2e_3 - b_2y_2x_1y_3 + y_2x_1a_3x_3 \\ & + y_1x_3b_2y_2 + y_1x_3a_2x_2 - y_1x_2b_3y_3 - y_1x_2a_3x_3 - y_2x_3b_1y_1 + a_1x_1x_2y_3, \end{aligned}$$

and

$$\begin{aligned} f'(t) = {} & -y_2x_3f_1 - y_2x_3c_1x_1 + y_1x_3d_2y_2 + y_1x_3c_2x_2 - y_1x_2c_3x_3 - y_1x_2d_3y_3 \\ & + c_1x_1x_2y_3 + y_2x_1f_3 - y_1x_2f_3 + y_1x_3f_2 + f_1x_2y_3 - f_2x_1y_3 \\ & + y_2x_1c_3x_3 - y_2x_3d_1y_1 + y_2x_1d_3y_3 + d_1y_1x_2y_3 - d_2y_2x_1y_3 - c_2x_2x_1y_3. \end{aligned}$$

Indeed, the transformation matrix

$$\begin{pmatrix} a(t) & b(t) \\ c(t) & d(t) \end{pmatrix}$$

is regular. Simplifying the expression $a(t)d(t) - b(t)c(t)$ gives the result

$$\frac{x_1(t)y_2(t) - x_2(t)y_1(t) - x_1(t)y_3(t) + x_3(t)y_1(t) + x_2(t)y_3(t) - x_3(t)y_2(t)}{y_2x_1 - y_3x_1 - y_2x_3 + y_1x_3 + y_3x_2 - y_1x_2},$$

where $x_i(t) = a_i(t)x_i + b_i(t)y_i + e_i(t)$ and $y_i(t) = c_i(t)x_i + d_i(t)y_i + f_i(t)$, $i = 1, 2, 3$. This denominator of this expression is zero if and only if the three points (x_1, y_1), (x_2, y_2) and (x_3, y_3) are collinear. By assumption, the points (x_1, y_1), (x_2, y_2) and (x_3, y_3) form a triangle, however. The numerator is non-zero since the points $g_1(x_1, y_1, t)$, $g_2(x_2, y_2, t)$ and $g_3(x_3, y_3, t)$ form a triangle S_t at any moment $t \in I$.

The coefficients of the resulting affine transformation are linear functions of the coefficients of the original transformations of the corner points (x_1, y_1), (x_2, y_2) and (x_3, y_3). As the original transformations are rational, the resulting affine transformation is rational too. □

Lemma 5.9. Let $\mathcal{O}_i = (\{(x_i, y_i)\}, I, g_i)$ $(i = 1, 2)$ be two atomic geometric objects with $g_i \in \mathcal{F}_{\text{Aff}}^{\text{Rat}}$. If the two points (x_1, y_1) and (x_2, y_2) form a line segment L (*i.e.*, are not equal) and if $g_1(x_1, y_1, t)$ and $g_2(x_2, y_2, t)$ form a line segment L_t at any moment $t \in I$ (*i.e.*, are not equal), then there exists an atomic geometric $\mathcal{O} = (L, I, g)$ with $g \in \mathcal{F}_{\text{Aff}}^{\text{Rat}}$ such that $g_i(x_i, y_i, t) = g(x_i, y_i, t)$ for all $t \in I$ and $i = 1, 2$.

Proof. Let (x_1, y_1) and (x_2, y_2) be the two end points of the line segment L and let (x_i, y_i) be transformed by the affinity g_i given by

$$\begin{pmatrix} a_i(t) & b_i(t) \\ c_i(t) & d_i(t) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i(t) \\ f_i(t) \end{pmatrix}, \quad i = 1, 2.$$

We prove that there always exists a rational affine functions $a(t)$, $b(t)$, $c(t)$ and $d(t)$, such that the matrix

$$\begin{pmatrix} a(t) & b(t) \\ c(t) & d(t) \end{pmatrix}$$

transforms the line segment as described in the statement of this lemma (so, the translation components $e(t)$ and $f(t)$ of this affinity are identical zero).

The condition for the existence of a single affinity that transforms the two endpoints of the line segment according to their respective affinities is that the first matrix in the following equation is regular.

$$\begin{pmatrix} x_1 & y_1 & 0 & 0 \\ 0 & 0 & x_1 & y_1 \\ x_2 & y_2 & 0 & 0 \\ 0 & 0 & x_2 & y_2 \end{pmatrix} \begin{pmatrix} a(t) \\ b(t) \\ c(t) \\ d(t) \end{pmatrix} = \begin{pmatrix} a_1(t)x_1 + b_1(t)y_1 + e_1(t) \\ c_1(t)x_1 + d_1(t)y_1 + f_1(t) \\ a_2(t)x_2 + b_2(t)y_2 + e_2(t) \\ c_2(t)x_2 + d_2(t)y_2 + f_2(t) \end{pmatrix}$$

This is true if the two endpoints of the line segment do not coincide.

The affinity that determines the movement of the intersection, can be found by solving the above equation. First, we abbreviate the expression $x_1y_2 - x_2y_1$ by D. The coefficients $a(t)$, $b(t)$, $c(t)$ and $d(t)$ are all of the form $\frac{a'(t)}{D}$, $\frac{b'(t)}{D}$, $\frac{c'(t)}{D}$ and $\frac{d'(t)}{D}$, respectively, where

$$a'(t) = e_1(t)y_2 - y_1a_2(t)x_2 - y_1b_2(t)y_2 - y_1e_2(t) + a_1(t)x_1y_2 + b_1(t)y_1y_2,$$

$$b'(t) = -x_2e_1(t) + x_1a_2(t)x_2 + x_1b_2(t)y_2 + x_1e_2(t) - x_2a_1(t)x_1 - x_2b_1(t)y_1,$$

$$c'(t) = f_1(t)y_2 - y_1c_2(t)x_2 - y_1d_2(t)y_2 - y_1f_2(t) + c_1(t)x_1y_2 + d_1(t)y_1y_2,$$

and

$$d'(t) = -x_2f_1(t) + x_1c_2(t)x_2 + x_1d_2(t)y_2 + x_1f_2(t) - x_2c_1(t)x_1 - x_2d_1(t)y_1.$$

As in the case of the previous lemma, it can be shown that

$$\begin{pmatrix} a(t) & b(t) \\ c(t) & d(t) \end{pmatrix}$$

is regular and therefore determines an affinity.

This solution is linear in the components of the original rational affine transformations of $\mathcal{O}_1$ and $\mathcal{O}_1$, so it is also rational. □

The next lemma shows that if two lines that move with a rational affinity intersect, also the intersection point is moved by a rational affinity.

Lemma 5.10. Let $\mathcal{O}_i = (L_i, I, g_i)$ $(i = 1, 2)$ be two atomic geometric objects with L_i line segments and $g_i \in \mathcal{F}_{\text{Aff}}^{\text{Rat}}$. If the line segments $g_1(L_1, t)$ and $g_2(L_2, t)$ intersect at any moment $t \in I$, then there exists an atomic geometric $\mathcal{O} = (\{(x_0, y_0)\}, I, g)$ with $g \in \mathcal{F}_{\text{Aff}}^{\text{Rat}}$ that describes the intersection point of $g_1(L_1, t)$ and $g_2(L_2, t)$ in I.

Proof. Let (x_i, y_i) and (u_i, v_i) be the two end points of the line segment L_i $(i = 1, 2)$. Let L_i be transformed by the affinity g_i given by

$$\begin{pmatrix} a_i(t) & b_i(t) \\ c_i(t) & d_i(t) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i(t) \\ f_i(t) \end{pmatrix}, \quad i = 1, 2.$$

We compute the intersection of $g_1(L_1, t)$ and $g_2(L_2, t)$ by solving the equations

$$\begin{aligned} &\lambda_1(a_1(t)x_1 + b_1(t)y_1 + e_1(t)) + (1 - \lambda_1)(a_1(t)u_1 + b_1(t)v_1 + e_1(t)) \\ &\qquad = \lambda_2(a_2(t)x_2 + b_2(t)y_2 + e_2(t)) + (1 - \lambda_2)(a_2(t)u_2 + b_2(t)v_2 + e_2(t)) \end{aligned}$$

and

$$\begin{aligned} &\lambda_1(c_1(t)x_1 + d_1(t)y_1 + f_1(t)) + (1 - \lambda_1)(c_1(t)u_1 + d_1(t)v_1 + f_1(t)) \\ &\qquad = \lambda_2(c_2(t)x_2 + d_2(t)y_2 + f_2(t)) + (1 - \lambda_2)(c_2(t)u_2 + d_2(t)v_2 + f_2(t)) \end{aligned}$$

in λ_1 and λ_2. The determinant of the matrix

$$\begin{pmatrix} a_1(t)x_1 + b_1(t)y_1 - a_1(t)u_1 - b_1(t)v_1 & a_2(t)u_2 + b_2(t)v_2 - a_2(t)x_2 - b_2(t)y_2 \\ c_1(t)x_1 + d_1(t)y_1 - c_1(t)u_1 - d_1(t)v_1 & c_2(t)u_2 + d_2(t) - v_2c_2(t)x_2 - d_2(t)y_2 \end{pmatrix}$$

is zero if one of the $g_i(L_i, t)$ is parallel to one of the coordinate axes or if both line segments are parallel. The latter case is no problem as we can use the finite time partition (Property 5.2.5) to consider only those subintervals J of I during which the intersection exists. We treat the case of line segments parallel to one of the coordinate axis separately.

If the line segments are not parallel to one of the coordinate axes, the intersection point is the following. We only give the x-coordinate $s_x(t)$ (the y-coordinate $s_y(t)$ is expressed similarly). For clarity time-dependence in the coefficients of the affinities is omitted.

We have that $s_x(t)((a_1x_1 + b_1y_1 - a_1u_1 - b_1v_1)(-d_2v_2 + c_2x_2 + d_2y_2 - c_2u_2) + (a_2u_2 + b_2v_2 - a_2x_2 - b_2y_2)(c_1x_1 + d_1y_1 - c_1u_1 - d_1v_1))$ equals

$$
\begin{aligned}
&((((x_2v_1 - u_2v_1)a_2 + (-v_2v_1 + y_2v_1)b_2)d_1 + ((f_1 - d_2v_2 - f_2)x_2 + u_2d_2y_2\\
&+(-f_1+f_2)u_2)a_2+(e_2c_2+b_2v_2c_2)x_2+((-f_2-c_2u_2+f_1)y_2+(-f_1+f_2)v_2)b_2+e_2d_2y_2\\
&- e_2d_2v_2 - e_2c_2u_2)x_1 + ((u_2u_1 - x_2u_1)a_2 + (-y_2u_1 + v_2u_1)b_2)d_1y_1 + ((u_1d_2v_2\\
&+(-f_1+f_2)u_1)x_2 - u_2u_1d_2y_2 + (f_1 - f_2)u_2u_1)a_2 + (-b_2v_2u_1c_2 - e_2u_1c_2)x_2 + ((u_1c_2u_2\\
&+ (-f_1 + f_2)u_1)y_2 + (f_1 - f_2)v_2u_1)b_2 - e_2u_1d_2y_2 + e_2u_1d_2v_2 + e_2u_1c_2u_2)a_1\\
&+(((-x_2v_1+u_2v_1)a_2+(-y_2v_1+v_2v_1)b_2)c_1b_1+((u_2e_1-x_2e_1)a_2+(v_2e_1-y_2e_1)b_2)c_1)x_1\\
&+ ((((-u_2u_1 + x_2u_1)a_2 + (-v_2u_1 + y_2u_1)b_2)c_1 + ((f_1 - d_2v_2 - f_2)x_2 + u_2d_2y_2\\
&+(-f_1+f_2)u_2)a_2+(e_2c_2+b_2v_2c_2)x_2+((-f_2-c_2u_2+f_1)y_2+(-f_1+f_2)v_2)b_2+e_2d_2y_2\\
&- e_2d_2v_2 - e_2c_2u_2)y_1 + ((v_1d_2v_2 + (-f_1 + f_2)v_1)x_2 - u_2v_1d_2y_2 + (f_1 - f_2)u_2v_1)a_2\\
&+ (-b_2v_2v_1c_2 - e_2v_1c_2)x_2 + ((v_1c_2u_2 + (-f_1 + f_2)v_1)y_2 + (f_1 - f_2)v_2v_1)b_2 - e_2v_1d_2y_2\\
&+ e_2v_1c_2u_2 + e_2v_1d_2v2)b_1 + ((u_2e_1 - x_2e_1)a_2 + (v_2e_1 - y_2e_1)b_2)d_1y_1\\
&+ ((x_2u_1e_1 - u_2u_1e_1)a_2 + (y_2u_1e_1 - v_2u_1e_1)b_2)c_1\\
&+ ((x_2v_1e_1 - u_2v_1e_1)a_2 + (-v_2v_1e_1 + y_2v_1e_1)b_2)d_1).
\end{aligned}
$$

For the intersection point to exists, $((a_1x_1 + b_1y_1 - a_1u_1 - b1_v1)(-d_2v_2 + c_2x_2 + d_2y_2 - c_2u_2) + (a_2u_2 + b_2v_2 - a_2x_2 - b_2y_2)(c_1x_1 + d_1y_1 - c_1u_1 - d_1v_1))$ should be different from zero. This condition expresses the fact that the line segments are not parallel, which is true by assumption.

The intersection point moves rationally, as its functions of time are rational functions in the coefficients of the original transformations. For any choice of reference point, it is clear that a rational affinity can be found that moves it as described by the above formulas $(s_x(t), s_y(t))$.

If one of the line segments $g_1(L_1, t)$ or $g_2(L_2, t)$ is parallel to the x-axis, the intersection point will have as y-coordinate the y-coordinate of that line segment. The same holds for segments parallel to the y-axis. In the case that one segment is parallel to the y-axis and the other to the x-axis, the intersection point moves with linear, polynomial, respectively rational functions of time, if both the objects $\mathcal{O}_1$ and $\mathcal{O}_2$ move with linear, polynomial, respectively rational functions of time. □

5.2.2.3 Results for Affinities

We can now start our series of closure and non-closure lemmas and start with the affine transformations. For the most general classes we have the following positive result.

Lemma 5.11 (Closure for $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$). Both the classes $\langle \mathcal{S}_{\mathrm{Poly}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ are closed under $\cap$ and $\setminus$.

Proof. By Property 5.2.4, it suffices to show this lemma for triangles. By Property 5.2.2 (*atomicity*) and Property 5.2.3, it suffices to show that the intersection of two atomic geometric objects $\mathcal{O}_1 = (T_1, I_1, f_1)$ and $\mathcal{O}_2 = (T_2, I_2, f_2)$ from the class $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ is represented by an object in $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$.

According to Property 5.2.5 (*finite time partition*), the intersection of the two moving triangles can only take a finite number of different shapes, with each new shape occurring in an element of a finite partition of $I_1 \cup I_2$ into intervals $J_1, \ldots, J_m$ (in fact, we only have to consider $I_1 \cap I_2$ here, since outside this intersection the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ is empty anyway). Let J_l be an interval in this partition. The intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ can be a convex polygon (with at most six corner points), a line segment or a single point in J_l.

First, suppose the intersection is a convex polygon. Let τ_0 be a point in J_l (even if it is a degenerated interval, J_l contains at least one point). We take the intersection of $f_1(T_1, \tau_0)$ and $f_2(T_2, \tau_0)$ as reference object P. The set $P \subset \mathbb{R}^2$ can be triangulated, for instance by connecting its corner points to its point of gravity: this yields triangles $T'_1, \ldots, T'_m$ (with $1 \leq m \leq 6$). Each of the corner points (x_1, y_1), (x_2, y_2), (x_3, y_3) of a triangle T'_j is moved in the time interval J_l by a rational affinity (in particular it is moved f_1 or f_2 applied to the inverse image of $f_1(\cdot, \tau_0)$, respectively $f_2(\cdot, \tau_0)$). More specifically, a corner point of T'_j is moved by f_1 if it is originating from a corner point of $\mathcal{O}_1$; a corner point of T'_j is moved by f_2 if it is originating from a corner point of $\mathcal{O}_2$; Lemma 5.10 shows that there exists a rational affinity that moves a corner point of T'_j if it is an intersection point of side lines of $\mathcal{O}_1$ and $\mathcal{O}_2$; a corner point of T'_j can be taken to be moved by f_1 if it is originating from the point of gravity of P. Therefore, all corner points of T'_j are moved by a rational affinity. Lemma 5.8 guarantees the existence of a rational affinity f_j that moves T'_j. The intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ in J_l is therefore described by the atomic geometric objects (T'_j, J_l, f_j) $(1 \leq j \leq m \leq 6)$.

Second, we investigate the situation if the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ is a line segment. The end points of the intersection originate from $\mathcal{O}_1$ or $\mathcal{O}_2$ or can be the result of intersecting side lines of $\mathcal{O}_1$ and $\mathcal{O}_2$. In both cases, (from Lemma 5.10 for an intersection point) it is clear that the two end points are moved by a rational affine transformation. Lemma 5.9 then shows that there exists a single rational affine transformation f to move the intersection. This intersection can therefore be described by an atomic geometric object (L, J_l, f), where L is some line segment.

Third, we look at the case where the intersection is a single point. This point can originate from $\mathcal{O}_1$ or $\mathcal{O}_2$ or can be the result of intersecting side lines of $\mathcal{O}_1$ and $\mathcal{O}_2$. In both cases, (from Lemma 5.10 for an intersection point), it is clear that in this case the intersection's movement is a rational affine transformation. □

In general, if the affine transformations of $\mathcal{O}_1$ and $\mathcal{O}_2$ are given by polynomial or linear functions, the corner points (x_1, y_1), (x_2, y_2) and (x_3, y_3) of triangles in the intersection (or difference) are in general rational in these functions. The computations in the proof of the Lemmas 5.8, 5.9 and 5.10 suggest that this leads to non-closure.

Lemma 5.12 (Non-closure for $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Pol}}_{\mathrm{Aff}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Lin}}_{\mathrm{Aff}} \rangle$). The classes $\langle \mathcal{S}_{\mathrm{Poly}}, \mathcal{F}^{\mathrm{Pol}}_{\mathrm{Aff}} \rangle$, $\langle \mathcal{S}_{\mathrm{Poly}}, \mathcal{F}^{\mathrm{Lin}}_{\mathrm{Aff}} \rangle$, $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Pol}}_{\mathrm{Aff}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Lin}}_{\mathrm{Aff}} \rangle$ are not closed under $\cap$ and $\setminus$.

Proof. It suffices to prove the lemma for triangles. We give a counterexample for intersection that serves for both classes $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Lin}}_{\mathrm{Aff}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Pol}}_{\mathrm{Aff}} \rangle$. Consider two atomic geometric objects $\mathcal{O}_1$ and $\mathcal{O}_2$ with reference objects triangles with corner points $(1, 1)$, $(3, 1)$, $(2, 3)$ and $(2, 2)$, $(4, 2)$, $(3, 4)$, respectively. The affine transformations of these triangles are given by the matrices

$$\begin{pmatrix} t & 2t \\ 3t & t \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} t & 2t+1 \\ t & 3t+1 \end{pmatrix},$$

respectively. Assume these objects are moved in some interval of the strictly positive t-axis (for example $I = [1, 2]$), the intersection of the two objects is a triangle with corner points $(6t + 2, 8t + 2)$, $(\frac{1}{2}t\frac{(181t+70)}{(13t+4)}, \frac{1}{2}t\frac{(243t+70)}{(13t+4)})$ and $(\frac{29}{4}t, \frac{37}{4}t)$.

Assume that this triangle could be represented as a geometric object $\{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$ from $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Pol}} \rangle$. Then, there exists some subinterval J of I during which the corner point $(\frac{1}{2}t\frac{(181t+70)}{(13t+4)}, \frac{1}{2}t\frac{(243t+70)}{(13t+4)})$ is the image of a corner point (x_0, y_0) of a reference triangle that is transformed by a polynomial (or linear) affinity. We therefore have that, for instance the x-coordinate $\frac{1}{2}t\frac{(181t+70)}{(13t+4)}$ of the above point is of the form $a(t)x_0 + b(t)y_0 + e(t)$ for $t \in J$ with $a(t)$, $b(t)$ and $e(t)$ polynomials (or linear polynomials) in t. Therefore, $181t^2 + 70t - 2(a(t)x_0 + b(t)y_0 + e(t))(13t + 4) = 0$ for all $t \in J$. Since the number of zero's of this polynomial exceeds its degree, it is identical to zero. Therefore, $a(t)x_0 + b(t)y_0 + e(t)$ is of the form $\alpha t + \beta$. This leads to the conditions $\beta = 0$, $181 = 26\alpha$ and $70 = 8\alpha$. There is no solution and we have a contradiction. □

Lemma 5.13 (Closure for $\langle \mathcal{S}_{\mathrm{Rect}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ and $\langle \mathcal{S}_{\mathrm{TrAx}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$). Both the classes $\langle \mathcal{S}_{\mathrm{Rect}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ and $\langle \mathcal{S}_{\mathrm{TrAx}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ are closed under $\cap$ and $\setminus$.

Proof. Let us first consider the class $\langle \mathcal{S}_{\mathrm{Rect}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$. Because of Lemmas 5.2.2, and 5.2.3, it suffices to consider the intersection of two atomic geometric objects $\mathcal{O}_1 = (R_1, I_1, f_1)$ and $\mathcal{O}_2 = (R_2, I_2, f_2)$. The image of a rectangle under an affinity is a parallelogram. The shape of the intersection of $f_1(R_1, t)$ and $f_2(R_2, t)$ for some t in $I_1 \cap I_2$ can therefore be a convex polygon with at most eight corner points, a line segment or a point. In any of these cases, we can copy the argumentation used in the proof of Lemma 5.11.

In case the intersection is a line segment or a point, this settles the case. In the case where it is a convex polygon, we can reuse the triangulation technique presented in the proof of Lemma 5.11, now noting that it can consist of at most eight triangles instead of six. So, we get that the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ can be described by the atomic geometric objects (T'_j, J_l, f_j) $(1 \leq j \leq m \leq 8)$, where the T'_j are triangles and the f_j are rational affinities.

For the purpose of this lemma, we need to describe the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ by means of moving rectangles, however. This can be achieved by replacing each of the triangles T'_j by three rectangles R_{1j}, R_{2j} and R_{3j}. Let the corner points of T'_j be (x_1, y_1), (x_2, y_2) and (x_3, y_3). The rectangle R_{ij} are chosen such that a constant affinity f_{ij} maps R_{ij} to the parallelogram with corner points (x_i, y_i), $\frac{1}{2}(x_1+x_2, y_1+y_2)$, $\frac{1}{2}(x_1 + x_3, y_1 + y_3)$ and $\frac{1}{2}(x_3 + x_2, y_3 + y_2)$ $(i = 1, 2, 3)$. So, T'_j is the union of the three parallelograms: $T'_j = f_{1j}(R_{1j}) \cup f_{2j}(R_{2j}) \cup f_{3j}(R_{3j})$.

So, if we replace (T'_j, J_i, f_j) by $(R_{ij}, J_l, f_j \circ f_{ij})$ we get a description of the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ during J_l in terms of atomic geometric objects from the class $\langle \mathcal{S}_{\mathrm{Rect}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$.

The closure result for $\langle \mathcal{S}_{\mathrm{TrAx}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ can be obtained by further dividing the rectangles $R_{i,j}$ along a diagonal into two triangles from $\mathcal{S}_{\mathrm{TrAx}}$. □

The following lemma concludes the results for affinities.

Lemma 5.14 (Non-closure for $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{L}}_{\text{Aff}} \rangle$ and $\langle \mathcal{S}_{\text{TrAx}}, \mathcal{F}^{\text{L}}_{\text{Aff}} \rangle$, $\text{L} \in \{\text{Lin}, \text{Poly}\}$). The classes $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{L}}_{\text{Aff}} \rangle$ and $\langle \mathcal{S}_{\text{TrAx}}, \mathcal{F}^{\text{L}}_{\text{Aff}} \rangle$ are both not closed under $\cap$ and $\setminus$ for $\text{L} \in \{\text{Lin}, \text{Poly}\}$.

Proof. First, let us look at $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{L}}_{\text{Aff}} \rangle$. We give a counterexample for intersection that serves for both classes $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{Lin}}_{\text{Aff}} \rangle$ and $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{Pol}}_{\text{Aff}} \rangle$. We modify the counterexample from the proof of Lemma 5.12. Consider two atomic geometric objects $\mathcal{O}_1$ and $\mathcal{O}_2$ with reference objects rectangles with corner points $(1,1)$, $(3,1)$, $(1,3)$, $(3,3)$ and $(2,2)$, $(4,2)$, $(2,4)$, $(4,4)$, respectively. The affine transformations of the rectangles are given by the matrices

$$\begin{pmatrix} t & 2t \\ 3t & t \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} t & 2t+1 \\ t & 3t+1 \end{pmatrix},$$

respectively.

In some interval of the strictly positive t-axis, the intersection of the two objects is a triangle with corner points $(6t+2, 8t+2)$, $(t\frac{(28t+15)}{(3t+2)}, 3t\frac{(13t+2)}{(3t+2)})$ and $(\frac{21}{2}t, \frac{17}{2}t)$.

The same type of argumentation as in the proof of Lemma 5.12, can be used to show that at least a rational affinity is needed to describe the intersection. Therefore, both $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{Lin}}_{\text{Aff}} \rangle$ and $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{Pol}}_{\text{Aff}} \rangle$ are not closed for intersection and set-difference.

Secondly, for $\langle \mathcal{S}_{\text{TrAx}}, \mathcal{F}^{\text{L}}_{\text{Aff}} \rangle$, we can reuse the above counterexample leaving out the corner points $(1,1)$ and $(4,4)$ respectively. The intersection remains the same and the argumentation can be repeated. □

The proof of Lemma 5.11 is based on the property that affinities do not preserve parallelism to the axes. We will see later that for scalings, which do preserve parallelism to the axes, the class of the objects of $\mathcal{S}_{\text{TrAx}}$ is not closed.

5.2.2.4 Results for Scalings

We divide the results for scalings into one positive and two negative results.

Lemma 5.15 (Closure for $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{L}}_{\text{Sc}} \rangle$, $\text{L} \in \{\text{Lin}, \text{Poly}, \text{Rat}\}$). $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}^{\text{L}}_{\text{Sc}} \rangle$ is closed under $\cap$ and $\setminus$ for $\text{L} \in \{\text{Lin}, \text{Poly}, \text{Rat}\}$.

Proof. Because of Lemmas 5.2.2, and 5.2.3, it suffices to consider the intersection of two atomic geometric objects $\mathcal{O}_1 = (R_1, I_1, g_1)$ and $\mathcal{O}_2 = (R_2, I_2, g_2)$.

According to Property 5.2.5, the intersection of the two rectangles takes different shapes in elements of a finite partition of $I_1 \cap I_2$ (we only consider this intersection, since elsewhere in $I_1 \,\overline{\cup}\, I_2$ the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ is empty in any case). Let J be an interval in this partition. First, we remark that scalings map lines that are parallel to the x-axis or to the y-axis to a parallel line. Therefore, at any moment t in J both the frame of $\mathcal{O}_1$ and the frame of $\mathcal{O}_2$ are rectangles with sides parallel to the coordinate axis.

Let us assume that the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ is a rectangle in J.

We remark that this intersection rectangle is uniquely determined by the coordinates of its upper-left corner point $(x_{ul}(t), y_{ul}(t))$ and the coordinates of the lower-right corner point $(x_{lr}(t), y_{lr}(t))$. Let assume the upper-left corner point of the intersection comes from $\mathcal{O}_1$ and the lower-right from $\mathcal{O}_2$ (possibly we have to work with

the upper-right and lower-left corners, but this is equivalent). Let the scaling of $\mathcal{O}_1$ be determined by $a_1(t)$, $b_1(t)$, $e_1(t)$, $f_1(t)$ and the one of $\mathcal{O}_2$ by $a_2(t)$, $b_2(t)$, $e_2(t)$, $f_2(t)$ (following the matrix notation of section 5.1.2).

The intersection is an atomic geometric object $\mathcal{O} = (R, J, f)$ composed as follows. The reference rectangle R has as upper-left corner point (x_{ul}, y_{ul}) the upper-left corner point of the reference object R_1 of $\mathcal{O}_1$ and as lower-right corner point (x_{lr}, y_{lr}) the lower-right corner point of the reference object R_2 of $\mathcal{O}_2$ (if (x_{lr}, y_{lr}) and (x_{lr}, y_{lr}) have an x- or y-coordinate in common, we work with $(x_{lr}+1, y_{lr}+1)$ instead of (x_{lr}, y_{lr}) and replace $e_2(t)$ with $e_2(t) - a_2(t)$ and $f_2(t)$ with $f_2(t) - b_2(t)$ in the description of g_2). The transformation function g of $\mathcal{O}$ is determined by

$$a(t) = \frac{(a_1(t)x_{ul} - a_2(t)x_{lr} + e_1(t) - e_2(t)}{x_{ul} - x_{lr}},$$

$$b(t) = \frac{(b_1(t)y_{ul} - b_2(t)y_{lr} + f_1(t) - f_2(t)}{y_{ul} - y_{lr}},$$

$$e(t) = \frac{((a_2(t) - a_1(t))x_{ul}x_{lr} - e_1(t)x_{lr} + e_2(t)x_{ul}}{x_{ul} - x_{lr}}, \text{ and}$$

$$f(t) = \frac{((b_2(t) - b_1(t))y_{ul}y_{lr} - f_1(t)y_{lr} + f_2(t)y_{ul}}{y_{ul} - y_{lr}}.$$

These formulas show that, if the transformations of $\mathcal{O}_1$ and $\mathcal{O}_2$ are rational, polynomial or linear, respectively, then also $a(t)$, $b(t)$, $e(t)$ and $f(t)$ are rational, polynomial and linear, respectively.

The cases where the intersection of $\mathcal{O}_1$ and $\mathcal{O}_2$ is a line segment or point in J are analogous to but simpler than the previous case. □

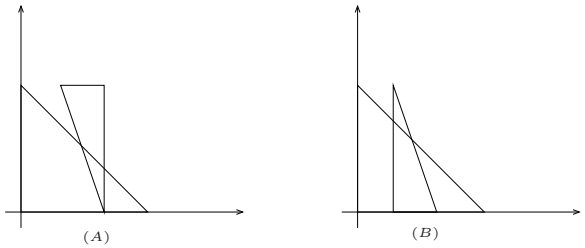


Figure 5.3: Counterexamples for intersection (A) and difference (B) for the classes $\langle \mathcal{S}_{\mathrm{TrAx}}, \mathcal{F}_{\mathrm{Sc}}^{\mathrm{Lin}} \rangle$

Lemma 5.16 (**Non-closure for $\langle \mathcal{S}_{\mathrm{TrAx}}, \mathcal{F}_{\mathrm{Sc}}^{\mathrm{L}} \rangle$ and $\langle \mathcal{S}_{\mathrm{TrAx}}, \{\mathrm{id}\} \rangle$**). The classes $\langle \mathcal{S}_{\mathrm{TrAx}}, \mathcal{F}_{\mathrm{Sc}}^{\mathrm{L}} \rangle$ and $\langle \mathcal{S}_{\mathrm{TrAx}}, \{\mathrm{id}\} \rangle$ are not closed under $\cap$ and $\setminus$ for $\mathrm{L} \in \{\mathrm{Lin}, \mathrm{Poly}, \mathrm{Rat}\}$.

Proof. Consider the triangle with corner points $(0,0)$, $(1,0)$ and $(0,1)$ and the triangle with corner points $(\frac{1}{3}, 1)$, $(\frac{2}{3}, 1)$ and $(\frac{2}{3}, 0)$, both transformed by the identity transformation. Their intersection (for an illustration see (A) of Figure 5.3) cannot be described as a finite union of elements of $\langle \mathcal{S}_{\mathrm{TrAx}}, \mathcal{F}_{\mathrm{Sc}}^{\mathrm{L}} \rangle$ since scalings map lines

that are parallel to a coordinate axis to a parallel line. (Remember, for affinities, this class was closed, partly because affinities do not necessarily preserve parallelism with the coordinate axis.) □

The following lemma could be left out since it is implied by Lemma 5.18. We nevertheless give it, since its proof is conceptually easier.

Lemma 5.17 (Non-closure for $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{L}}_{\mathrm{Sc}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Poly}}, \mathcal{F}^{\mathrm{L}}_{\mathrm{Sc}} \rangle$, $\mathrm{L} \in \{\mathrm{Lin}, \mathrm{Poly}\}$). The classes $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{L}}_{\mathrm{Sc}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Poly}}, \mathcal{F}^{\mathrm{L}}_{\mathrm{Sc}} \rangle$ are not closed under $\cap$ and $\setminus$ for $\mathrm{L} \in \{\mathrm{Lin}, \mathrm{Poly}\}$.

Proof. Because of Properties 5.2.2 and 5.2.4 it suffices to prove this for atomic geometric objects that have a triangle as a reference object. Consider the triangle with corner points $(0,0)$, $(0,1)$ and $(1,0)$, and the triangle with corner points $(0,0)$, $(1,1)$ and $(1,0)$. Their respective transformation functions are the scalings

$$\begin{pmatrix} 1 & 0 \\ 0 & t+1 \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} 1 & 0 \\ 0 & 2t+1 \end{pmatrix}.$$

We consider for both objects the time interval $[0,5]$. At any moment during this interval the intersection is given by the triangle with corner points $(0,0)$, $(\frac{t+1}{3t+2}, \frac{(t+1)(2t+1)}{3t+2})$ and $(1,0)$. Assume that this intersection is described by a geometric object $\{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$ from $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{L}}_{\mathrm{Sc}} \rangle$. At least one of the atomic objects describes a moving triangle that contains $(\frac{t+1}{3t+2}, \frac{(t+1)(2t+1)}{3t+2})$ as a corner point during some subinterval of $[0,5]$. The x-coordinate $\frac{t+1}{3t+2}$ is therefore of the form $a(t)x_0 + e(t)$ with x_0 the x-coordinate of some corner point of a reference object, and $a(t)$ and $e(t)$ functions appearing in its transformation matrix. Therefore, $a(t)x_0 + e(t)$ has degree 0, *i.e.*, it is a number, say α. But then $\alpha(3t+2)$ and $t+1$ should be identical polynomials, leading to the equations $3\alpha = 1$ and $2\alpha = 1$ that clearly do not have a solution. It can therefore not be a linear or polynomial transformation. □

The next lemma completes the proofs for scalings.

Lemma 5.18 (Non-closure for $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Rat}}_{\mathrm{Sc}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Poly}}, \mathcal{F}^{\mathrm{Rat}}_{\mathrm{Sc}} \rangle$). The classes $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Rat}}_{\mathrm{Sc}} \rangle$ and $\langle \mathcal{S}_{\mathrm{Poly}}, \mathcal{F}^{\mathrm{Rat}}_{\mathrm{Sc}} \rangle$ are not closed under $\cap$ and $\setminus$.

Proof. Because of Properties 5.2.2 and 5.2.4 it suffices to prove this lemma for atomic geometric objects that have a triangle as a reference object. We give an example of two atomic geometric objects $\mathcal{O}_1$ and $\mathcal{O}_2$ that have an intersection that cannot be described in $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Rat}}_{\mathrm{Sc}} \rangle$.

Let the reference triangle of the atomic geometric object $\mathcal{O}_1$ have corner points $(0,0)$, $(1,1)$ and $(\frac{1}{3}, \frac{1}{2})$ and let the transformation of this object be the scaling that maps (x,y) to

$$\begin{pmatrix} -\frac{3(t+1)}{t+3} & 0 \\ 0 & -(t+1) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} \frac{3(t+1)}{t+3} \\ t+1 \end{pmatrix}.$$

Let the reference triangle of the atomic geometric object $\mathcal{O}_2$ have corner points $(0,0)$, $(1,1)$ and $(0,1)$ and let the scaling of this object be the time-independent mapping that maps (x,y) to

$$\begin{pmatrix} 4 & 0 \\ 0 & 4 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} -2 \\ -2 \end{pmatrix}.$$

We consider both objects in the time interval $(0, \frac{1}{2})$. At any moment during this interval the intersection is given by the triangle with corner points $(0,0)$, $(\frac{3(t+1)}{t+3}, t+1)$ and $(1,1)$. We remark that the point $(\frac{3(t+1)}{t+3}, t+1)$ is situated above the diagonal $y = x$ and that in the limit towards 0, this point converges to $(1,1)$. In other words, the intersection is always a triangle during the time interval $(0, \frac{1}{2})$, but it converges to a line segment for t going to 0. It is easily verified that this intersection cannot be described as the image of a single triangle under a scaling from $\mathcal{F}_{\mathrm{Sc}}^{\mathrm{Rat}}$.

More generally, assume that this intersection is described by a geometric object $\{\mathcal{O}_1, \ldots, \mathcal{O}_m\}$ from $\langle\ \mathcal{S}_{\mathrm{Tr}},\ \mathcal{F}_{\mathrm{Sc}}^{\mathrm{Rat}}\ \rangle$. At least one of the atomic objects describes a moving triangle that covers a line segment connecting $(0,0)$ and $(f(t), f(t))$ of the line connecting $(0,0)$ and $(1,1)$ during a time interval $(0, \varepsilon]$ with $\varepsilon > 0$ (without loss of generality this interval can be assumed to be closed on the right side). Let the third cornerpoint $(g(t), h(t))$ be situated in the interior of the intersection triangle with cornerpoints $(0,0)$, $(\frac{3(t+1)}{t+3}, t+1)$ and $(1,1)$. Let the scaling of this object be the one that maps (x, y) to

$$\begin{pmatrix} a(t) & 0 \\ 0 & b(t) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} c(t) \\ d(t) \end{pmatrix},$$

where $(a(t)$, $b(t)$, $c(t)$ and $d(t)$ are rational functions of t. Without loss of generality the reference triangle of this atomic object can be assumed to have cornerpoints $(0,0)$, $(1,1)$, and (a,b), where the first is mapped to $(0,0)$, the second to $(f(t), f(t))$ and the third to $(g(t), h(t))$. Since we assume this reference object to be a triangle, we have $a \neq b$. It then follows that $a(t)$ and $b(t)$ must be equal to $f(t)$ and that $c(t)$ and $d(t)$ must be constant 0. Therefore, this scaling maps the third cornerpoint (a,b) to $(g(t), h(t)) = (af(t), bf(t))$. Both a and b are therefore strictly positive. Since the point $(af(t), bf(t))$ is situated at the same side of the diagonal $y = x$ as the point $(0,1)$, we get the condition $bf(t) - af(t) > 0$, or $b > a$. On the other hand, this point is situated on the same side as $(0,1)$ of the line connecting $(0,0)$ and $(\frac{3(t+1)}{t+3}, t+1)$. Therefore, we get

$$(t+1)af(t) - \frac{3(t+1)}{t+3} bf(t) > 0.$$

From this $a - \frac{3}{t+3} b > 0$ follows, or $\frac{3b}{a} < t+3$. Since $t \mapsto t+3$ is strictly increasing in $(0, \varepsilon]$ and has infimum 3 over this interval, we get $\frac{3b}{a} \leq 3$, or $b \leq a$. This contradicts $b > a$, that we obtained before. This concludes the proof. □

5.2.2.5 Results for Translations

We give a general negative result for translations.

Lemma 5.19 (**Non-closure for translations**)**.** For each of the classes $\mathcal{S}$ considered in the previous section, the class $\langle \mathcal{S},\ \mathcal{F}_{\mathrm{Trans}}^{\mathrm{L}}\ \rangle$ is not closed under $\cap$ and $\setminus$, for $\mathrm{L} \in \{\mathrm{Lin}, \mathrm{Poly}, \mathrm{Rat}\}$.

Proof First, we remark that translations preserve the shape and area of objects and the length of lines. Consider now two reference objects, located in the plane

$t = 0$, from each of the relevant classes that have the interval $[0, 1]$ on the x-axis as one of their sides. Let one reference object be located above the x-axis and the second be located below the x-axis. Let the first object undergo the translation $(x, y) \mapsto (x - t, y)$ in the direction of the negative x-axis and let the second object undergo the translation $(x, y) \mapsto (x + t, y)$ in the opposite direction, both in the time interval $[0, t_0]$, for some $t_0 > 0$. Then it is clear that the intersection of these objects is a shrinking line segment during the time interval $[0, t_0]$. So, in any of the cases, the intersection cannot be described as a finite union of translated objects. □

5.2.2.6 Results for the Identity

For completeness, we also give the results for the identity mapping.

Lemma 5.20. The classes $\langle \mathcal{S}_{\text{Poly}}, \mathcal{F}_{\text{id}} \rangle$, $\langle \mathcal{S}_{\text{Tr}}, \mathcal{F}_{\text{id}} \rangle$ and $\langle \mathcal{S}_{\text{Rect}}, \mathcal{F}_{\text{id}} \rangle$ are closed under $\cap$ and $\setminus$. The class $\langle \mathcal{S}_{\text{TrAx}}, \mathcal{F}_{\text{id}} \rangle$ is not closed under $\cap$ and $\setminus$.

Proof. For the positive closure results, it suffices to remark the following. The intersection of two polygons is again a polygon (if line segments and points are considered to be in this class). The intersection of two triangles is a convex polygon with at most six corner points that can be triangulated, *i.e.*, written as a disjoint union of triangles. The intersection of two rectangles is a rectangle, a line segment parallel to a coordinate axis, or a point.

For the negative result, we remark that the intersection of two reference objects from $\mathcal{S}_{\text{TrAx}}$ cannot necessarily be written as a finite union of such objects. Figure 5.3 contains an example. □

Now we have proven all the closure and non-closure results listed in the table of Theorem 5.5.

5.3 The Extended Data Model

It is clear that the model for representing spatio-temporal data, that we have presented in Section 5.1, gives mostly negative closure results (see Theorem 5.5) for the classes of objects we considered important for spatio-temporal practice. The only classes that seem to be useful for further investigation are $\langle \mathcal{S}, \mathcal{F}_{\text{Aff}}^{\text{Rat}} \rangle$, for any of the considered classes $\mathcal{S}$ of reference objects.

In this section, we will enrich the data model and get better closure results. We will also study normal forms for objects in this enriched model.

In Section 5.1, we defined a geometric object as a finite union of atomic objects. We could now try to modify this definition by allowing other operations than union in the construction of geometric objects from atomic geometric objects. The exhaustive list of alternative definitions that could be considered are: a geometric object is obtained from atomic geometric objects by means of

(a) union (see Section 5.2);

(b) intersection;

(c) set-difference;

(d) intersection and set-difference; and finally

(e) union, intersection and set-difference.

In this chapter, we will not investigate alternatives (b), (c) and (d). These alternatives may be interesting from a mathematical point of view, but in any practical application it is natural to allow union in the construction of spatio-temporal objects. In fact it is easy to see that for instance alternative (b) gives even worse closure results. Hereto, we first make two basic observations. Firstly, it is clear that the intersection of convex objects always results in a convex object, and that the affine transformation of a convex object remains a convex object. Secondly, the intersection of connected convex objects is again connected. It should be clear therefore that when the reference objects are triangles or rectangles, then whenever a union has two connected components, it cannot be written as an intersection of atomic geometric objects.

For alternative (c), we remark that in contrast to the intersection, the difference of two convex objects can result in a non-convex object, or in a set of disjoint objects. So, it is possible to describe a wide class of objects as the difference of some atomic objects. But, this approach has two major drawbacks:

1. If we want to describe a certain object $\mathcal{O}$ as the difference of some other objects $\mathcal{O}_1 \ldots \mathcal{O}_k$, we have to artificially introduce those objects $\mathcal{O}_1, \ldots, \mathcal{O}_k$ into the database. There is no way of controlling the number of objects that have to be introduced, as this depends on the exact shape of the object $\mathcal{O}$.

2. The difference operator is not associative, so in the worst case the depth of the tree describing the relation between the objects equals the number of objects. For practical applicability of our model, we should have a tree with limited depth. (One way of achieving this is to define a normal form, see further).

Only alternative (e) will be further investigated here.

5.3.1 The Extended Data Model

First, we define the extended model. Atomic geometric objects are defined as in Section 5.1.

Definition 5.21 (Extended data model)**.** An *extended geometric object* is a binary tree, where each non-leaf node has two children, where each of the nodes is labeled with $\cup$, $\cap$ or $\setminus$ and where each leaf is labeled with an atomic geometric object.

The semantics of a geometric object is defined (recursively starting from the root of the tree) as the semantics of its root. If a node n of the binary three has a left child lc and a right child rc, and if the root is labeled θ (with $\theta \in \{\cup, \cap, \setminus\}$), the semantics $sem(n)$ of node n is by definition $sem(lc)\ \theta\ sem(rc)$. The semantics of a leaf labeled with the atomic geometric object $\mathcal{O}$ is $st(\mathcal{O})$.

We define the *time domain* of an extended geometric object to be the convex closure of the union of the time domains of all the composing atomic geometric objects.

By slight abuse of notation, we will write down binary trees as in Definition 5.21 in the usual set-theoretic notation. The expression $\mathcal{O}_1 \cup (\mathcal{O}_2 \cap (\mathcal{O}_3 \setminus \mathcal{O}_1))$ is an example.

The following property is trivial and says that this model is closed for all Boolean set operations.

Property 5.3.1. For all the classes $\langle \mathcal{S}, \mathcal{F} \rangle$ considered in Section 5.1.2 the extended version of the data model is closed for union, intersection and set-difference.

5.3.2 Normal Forms for CSG

By allowing geometric objects to be constructed from atomic objects via union, intersection and difference, we arrive at a situation that is similar to what is used in the field of "Constructive Solid Geometry" (CSG) [45]. This is a method of geometric modeling, where complex static objects are constructed out of simple objects by taking the union, intersection and difference.

Looking at literature on CSG, we find that there exists a normal form for objects composed as Boolean combinations (with the operators $\cup$, $\cap$, $\setminus$) from atomic objects.

A tree representing a complex object (called a *CSG tree*) is in *normal form* when all intersection and subtraction operators have a left subtree which contains no union operators and a right subtree which is simply a primitive (a set of polygons representing a single solid object). All union operators are pushed towards the root, and all intersection and subtraction operators are pushed towards the leaves. In our setting, the primitives are atomic geometric objects and the complexes are geometric objects.

A CSG tree can be converted to normal form by repeatedly applying the following set of rewrite rules (which have the Church-Rosser property) to the tree and then its subtrees:

$$
\begin{array}{lcll}
A \setminus (B \cup C) & \rightsquigarrow & (A \setminus B) \setminus C & \text{(Rule 1)} \\
A \cap (B \cup C) & \rightsquigarrow & (A \cap B) \cup (A \cap C) & \text{(Rule 2)} \\
A \setminus (B \cap C) & \rightsquigarrow & (A \setminus B) \cup (A \setminus C) & \text{(Rule 3)} \\
A \cap (B \cap C) & \rightsquigarrow & (A \cap B) \cap C & \text{(Rule 4)} \\
A \setminus (B \setminus C) & \rightsquigarrow & (A \setminus B) \cup (A \setminus C) & \text{(Rule 5)} \\
A \cap (B \setminus C) & \rightsquigarrow & (A \cap B) \setminus C & \text{(Rule 6)} \\
(A \setminus B) \cap C & \rightsquigarrow & (A \cap C) \setminus B & \text{(Rule 7)} \\
(A \cup B) \setminus C & \rightsquigarrow & (A \setminus C) \cup (B \setminus C) & \text{(Rule 8)} \\
(A \cup B) \cap C & \rightsquigarrow & (A \cap C) \cup (B \cap C) & \text{(Rule 9)}
\end{array}
$$

where A, B, and C here can be both primitives or subtrees.

5.3.3 Normal Forms for Geometric Objects

First, we define the notion of normal form for a geometric object in the extended data model.

Definition 5.22 (Normal form). We say that a *geometric object* (in the extended version) is in *normal form* if every $\cap$- or $\setminus$-labeled node has no $\cup$-labeled node in the left subtree and has a right child that is labeled by an atomic object.

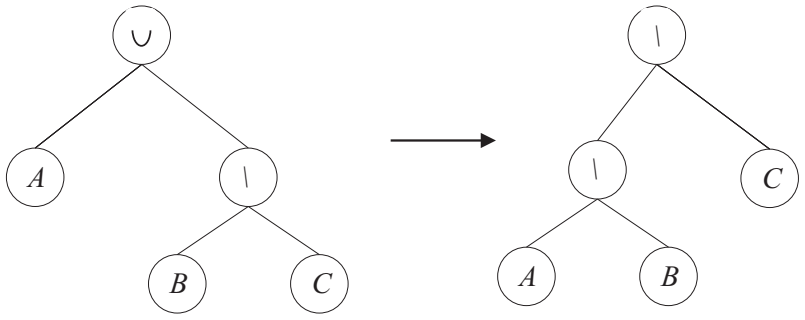


Figure 5.4: Tree notation for Rule 1. A, B, and C denote arbitrary subtrees. The arrow indicates how a subtree can be replaced by another subtree.

By Rule 7, differences can be pushed down with respect to intersections and we obtain, in the set-theoretic notation, that a geometric object is in normal form if it is of the form

$$\bigcup_{i=1}^{n}((\mathcal{O}_{i,1} \cap \cdots \cap \mathcal{O}_{i,k_i}) \setminus \mathcal{O}_{i,k_i+1} \setminus \cdots \setminus \mathcal{O}_{i,k_i+l_i})$$

where $\mathcal{O}_{i,j}$ is an atomic object.

The rewrite Rules 1–9 can be easily converted to tree notation, as illustrated for Rule 1 in Figure 5.4. The following property says that any geometric object can be rewritten in normal form. For the proof, we refer to [33].

Property 5.3.2. Any geometric object in the extended data model can be rewritten, using Rules 1–9, into a geometric object with the same semantics that is in normal form. Furthermore, this system of rewrite rules has the Church-Rosser property.

6

Triangulated Spatial and Spatio-temporal Data

In the previous chapter (Chapter 5), we modelled spatio-temporal objects as triples (S, I, f), where S is a spatial reference object in $\mathbb{R}^2$, I a time interval and f a transformation function which is, in general, a time-dependent affine transformation. We investigated the closure properties of several classes of spatio-temporal objects, and concluded that the class $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$ was interesting with respect to applications, as it is both the most general and closed under Boolean set operations.

However, a drawback of modelling an object as an arbitrary set of atomic objects, is that it is not clear immediately how the spatio-temporal object represented by the atomic objects looks like. Its time domain has to be computed from the time domains of all atomic objects, which might overlap. Also, there may be gaps, *i.e.*, moments when the spatio-temporal object does not exist, and two sets of atomic objects can represent the same spatio-temporal object. Or, there might be elements in the set of atomic objects that do not contribute to the spatio-temporal object at all, as they are overlapped totally by other atomic objects. In short, the model proposed would benefit from a normal form that supports visualization and describes the objects in a unique way.

The normal form that we develop will also be affine-invariant. The importance of affine-invariance is generally recognized in the computer graphics, robotics and computer vision communities. This is reflected in the widely-adopted *weak perspective assumption*. This is the assumption that when an object is repeatedly photographed from different viewpoints, and the object is relatively far away form the camera, that all pictures of the object are affine images of each other. We generalize this assumption for spatio-temporal objects as follows. If a spatio-temporal event is filmed by two moving observers, relatively far away from the event, then both films will be the same

up to a time-dependent affinity. For each time moment, another affinity maps the snapshots of the different movies onto each other.

As the objects we will consider have triangles as reference objects, we further refer to the normal form as a *triangulation*. We choose to define a spatial triangulation as a "partition" of the plane into triangles, that are allowed to share boundaries with each other. This is definition is commonly used (e.g. [28]). We will define a spatio-temporal triangulation later.

For spatial data, there exist several triangulation algorithms, but, apart from the triangulation of Nielson [56], their output is not affine-invariant. The method proposed by Nielson to triangulate a set of points in an affine-invariant way computes an affine-invariant norm using the coordinate information of all points, and then uses this norm in the triangulation algorithm. In Section 6.2, we will develop a spatial triangulation algorithm, that is more intuitive, is efficiently computable and naturally extends to a spatio-temporal triangulation algorithm. Afterwards, in Section 6.3, we develop a spatio-temporal triangulation method, based on this new spatial triangulation algorithm. We start with some definitions.

6.1 Affine-invariant Triangulation Methods

We first define the notions of spatial and spatio-temporal affine triangulation. We use the definition of spatio-temporal objects and geometric objects from Chapter 5 (see Definitions 5.1, 5.2 and 5.3).

Definition 6.1 (**Spatial and spatio-temporal triangulation**). Let $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ be a geometric object and τ_0 be a time moment in the time domain of $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$.
• A collection of triangles[1] $\{T_1, T_2, \ldots, T_m\}$ in $\mathbb{R}^2$ is a *triangulation* of the snapshot $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}^{\tau_0}$ if the interiors[2] of different T_i are disjoint and the union $\cup_{i=1}^{m} T_i$ equals $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}^{\tau_0}$.

• A geometric object $\{\mathcal{T}_1, \ldots, \mathcal{T}_m\}$ is a *triangulation of a geometric object* $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ if for each τ_0 in the time domain of $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$, $\{\mathcal{T}_1, \ldots, \mathcal{T}_m\}^{\tau_0}$ is a triangulation of the snapshot $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}^{\tau_0}$ and if $st(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}) = st(\{\mathcal{T}_1, \ldots, \mathcal{T}_m\})$.

We remark that in the second part of Definition 6.1, at each moment τ_0, the $\mathcal{T}_i^{\tau_0}$ are allowed to be empty (*i.e.*, τ_0 may be outside the time domain of $\mathcal{T}_i$).

In Figure 6.1, two stars with their respective triangulations are shown. Note that, although triangulations of spatial sets intuitively are *partitions* of such sets into triangles, they are not partitions in the mathematical sense. Indeed, the elements of the *partition* may have common boundaries. For spatial data, it is customary to allow the elements of a partition to share boundaries (see for example [28]).

A *spatial triangulation method* is a procedure that on input (some representation of) a snapshot of a spatio-temporal object produces (some representation of) a triangulation of this snapshot. A spatio-temporal *triangulation method* is a procedure that

[1] Remark that we consider filled triangles and we allow a triangle to degenerate into a closed line segment or a point.

[2] We define the *interior* as follows: the interior of a triangle is its topological interior; the interior of a line segment is the segment without endpoints; and the interior of a point is the point itself.

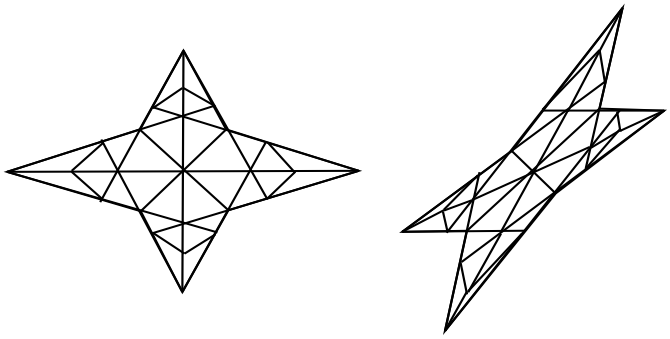


Figure 6.1: The triangulations of a snapshot (left) and of an affine transformation of the snapshot (right).

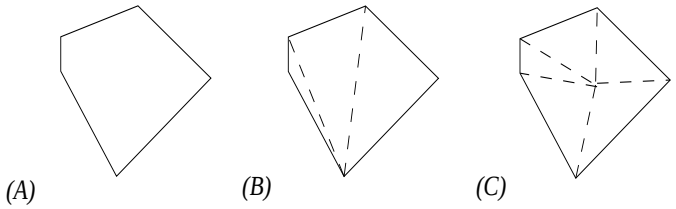


Figure 6.2: Two different triangulations ((*B*) and (*C*)) of a convex polygon ((*A*)).

on input (some representation of) a geometric object, produces (some representation of) a triangulation of this geometric object.

Next, we define what it means for such methods to be affine-invariant.

Definition 6.2 (**Affine-invariant triangulation methods**)**.** A spatial triangulation method $\mathcal{T}_S$ is called *affine invariant* if and only if for any two snapshots A and B, for which there is an affinity $\alpha : \mathbb{R}^2 \to \mathbb{R}^2$ such that $\alpha(A) = B$, also $\alpha(\mathcal{T}_S(A)) = \mathcal{T}_S(B)$.

A spatio-temporal triangulation method $\mathcal{T}_{ST}$ is called *affine invariant* if and only if for any geometric objects $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ and $\{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\}$ for which for each moment τ_0 in their time domains, there is an affinity $\alpha_{\tau_0} : \mathbb{R}^2 \to \mathbb{R}^2$ such that if $\alpha_{\tau_0}(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}^{\tau_0}) = \{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\}^{\tau_0}$, also $\alpha_{\tau_0}(\mathcal{T}_{ST}(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\})^{\tau_0}) = \mathcal{T}_{ST}(\{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\})^{\tau_0}$.

Example 6.3. Given a convex polygon as shown in Part (*A*) of Figure 6.2. A spatial triangulation method that takes the leftmost of the corner points with the smallest y-coordinates of the polygon and connects it with all other corner points, is not affine invariant. It is not difficult to see that, when an affine transformation is applied to the polygon, another point may become the leftmost lowest corner point. Part (*B*) of Figure 6.2 shows the result of applying this triangulation method to the convex polygon shown in Part (*A*).

A triangulation method that computes the barycenter of a convex polygon and connects it with all corner points is affine-invariant. An illustration is shown in Part (*C*) of Figure 6.2. □

We now propose an affine-invariant spatial triangulation method for spatial figures that are snapshots of geometric objects, or, that can be represented as finite sets of triangles.

6.2 An Affine-invariant Spatial Triangulation

We next propose an affine-invariant triangulation method. Later on, in Section 6.3, we will use the technique proposed here to construct a spatio-temporal triangulation algorithm. We first explain the intuition behind the triangulation method, and then give the details in Algorithm 3. We illustrate the algorithm with an example, prove its correctness and end with determining the size of the output and the time complexity of the algorithm.

Intuitively, the algorithm is as follows. The input is a snapshot S, given as a finite set of triangles. In Figure 6.3 (A), for example, a snapshot of a house-like shape is given by four triangles. One of those triangles is degenerated into a line segment (representing the chimney). To make sure that the triangulation is independent of the exact representation of the snapshot by means of triangles, the boundary of the snapshot, *i.e.*, the boundary of the union of the triangles composing S, is computed. For the snapshot of Figure 6.3, the boundary is shown in Part (B). The (triangle degenerated into a) line segment contributes to the boundary. Therefor, we label it, the reason for this will become clear in a further stage of the procedure. Also, the (triangles degenerated into) points of the input that are not part of a line segment or real triangle, *i.e.*, the ones contributing to the boundary, are added to the output immediately.

The set of all lines through the edges of the boundary partitions the plane into a set of open convex polygons, open line segments, open (half-) lines and points. The (half-) lines and some of the polygons can be unbounded, so we use the convex hull $\mathcal{CH}(S)$ of the corner points of all triangles in the input as a bounding box. In Part (C) of Figure 6.3, the grey area is the area inside of the convex hull. The partition of the area inside the convex hull is computed. The points in this partition are not considered. The points contributing to the boundary were already added to the output in an earlier stage. For each open line segments, it is checked whether it is part of a labelled line segment of the input. Recall that only line segments that contribute to the boundary are labelled in an earlier stage of the algorithm. Only if an open line segment is part of a labelled segment, as is the case for the one printed in bold in Figure 6.3 Part (D), its closure (*i.e.*, a closed line segment) is added to the output. For each open polygon in the partition, we compute the polygon that is its closure and triangulate this polygon using its center of mass (see Figure 6.3 Part (D) for a polygon in the partition and Part (E) for its triangulation). Some open polygons are only part of the convex hull of S, but not of the snapshot itself. The polygons shaded in grey in Part (D) of Figure 6.3 are an example of such polygons. If a polygon does not belong to S, we do not triangulate it. The triangulations of all other polygons are added to the output. Note that we can decide whether a polygon belongs to the snapshot by first computing the *planar subdivision* $\mathcal{U}(S)$ (which we will define next) of the input snapshot and then test for each open polygon whether its center of mass

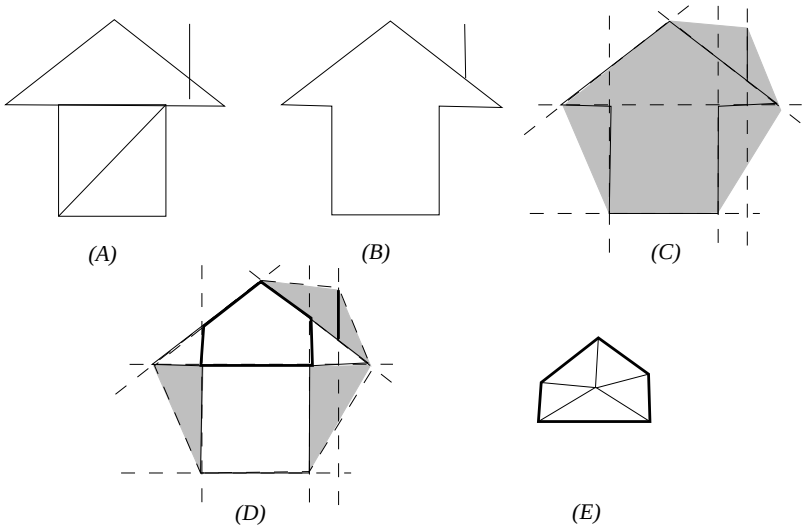


Figure 6.3: The several steps in the spatial triangulation algorithm.

belongs to the interior of a region or face in the subdivision. We will explain this in more detail when analyzing the complexity of the algorithm.

In the detailed description of the algorithm, we will use some well known techniques. One of those is the *doubly-connected edge list* [18], used to store *planar subdivisions*.

Definition 6.4 (**Planar subdivision**)**.** A *planar subdivision* is a subdivision of the plane into labelled regions (or *faces*), edges and vertices, induced by a plane graph. The *complexity of a subdivision* is the sum of the number of vertices, the number of edges and the number of faces it consists of.

Next, we describe the *doubly-connected edge list*, a data structure to store planar subdivisions. For this structure, each edge is split into two directed *half-edges*. In general, a doubly-connected edge list contains a record for each face, edge and vertex of the planar subdivision.

Definition 6.5 (**Doubly-connected edge list**)**.** Given a planar subdivision $\mathcal{S}$. A *doubly-connected edge list for* $\mathcal{S}$, denoted DCEL($\mathcal{S}$), is a structure containing a record for each face, edge and vertex of the subdivision. These records store the following geometric and topological information:

(i) The *vertex record* of a vertex **a** stores the coordinates of **a** and a pointer to an arbitrary half-edge that has **a** as its origin;

(ii) The *face record* of a face f stores a pointer to an arbitrary half-edge on its boundary. Furthermore, for each hole in f, it stores a pointer to an arbitrary half-edge on its boundary;

(iii) The *half-edge record* of a half-edge e stores five pointers. One to its origin-vertex, one to its twin half-edge, one to the face it bounds, and one to the previous and next half-edge on the boundary on that face.

Example 6.6. Figure 6.4 shows a planar subdivision (Part (B)) and its topological structure (Part (C)), that is reflected in the doubly-connected edge list represented in Table 6.1. □

Algorithm 3 (or $\mathcal{T}_S$) gives the triangulation procedure more formally. The input of this triangulation algorithm is a snapshot S, consisting of a geometric object which we assume to be given as a finite set of (possibly overlapping and possibly degenerated) triangles. We further assume that each triangle is represented as a triple of pairs of coordinates, which are rational numbers.

To shorten and simplify the exposition of Algorithm 3, we assume that S is fully two-dimensional, or equivalently, that points and line segments that are not adjacent to a polygon belonging to S are already in the output. Including their triangulation in the algorithm would make its description tedious, as we would have to add, and consider, more node and edge labels.

We use C programming-style notation for pointers to records and elements of records. For example, Let $\mathbf{a} = (a_1, a_2)$. In the vertex record V_a of $\mathbf{a}$, $V_a.x = a_1$ and $V_a.y = a_2$. Let e be an edge record. The coordinates of the origin $e \rightarrow$ origin of e are $e \rightarrow \text{origin} \rightarrow x$ and $e \rightarrow \text{origin} \rightarrow y$.

Before proving the correctness of the algorithm and determining the size of the output and the time complexity of the algorithm, we give an example.

Example 6.7. Let S be the set $\{T_1, T_2\}$, where T_1 is the triangle with corner points $v1$, $v3$ and $v4$, and T_2 the triangle with corner points $v2$, $v3$ and $v4$, as depicted in Figure 6.4. The doubly-connected edge list corresponding to Part (C) is shown in Table 6.1. We omitted the structures for vertices and faces, as we don't need them for the second part of the algorithm.

After the doubly-connected edge list is constructed, we create and output the triangles. This is done by visiting all half-edges once. Suppose we start with $e_{2,1}$. The next-pointers lead to $e_{1,5}$ and $e_{5,2}$. The next pointer of the last one points to $e_{2,1}$, which we visited already. This means we visited all edges of one polygon. The center of mass can now be computed and the triangles added to the output. This is done for all polygons that are part of the input. In this example, the polygon with corner points $v1$, $v5$ and $v2$ will not be triangulated, as it is not part of the input. The algorithm stops when there are no more unvisited edges left. □

Note that, as an optimization, we could decide to not triangulate faces that are triangles already. This does not influence the complexity results, however. Therefor, and also for a shorter and more clear exposition, we formulated the algorithm in a more general form.

We now prove compute the complexity of both the output and execution time of the triangulation method described in Algorithm 3 and afterwards show that it is affine-invariant. First, we show that $\mathcal{T}_S$ is indeed a triangulation method.

Algorithm 3 $\mathcal{T}_S$ (Input: $S = \{T_1, T_2, \ldots, T_k\}$, Output: $\{T'_1, T'_2, \ldots, T'_\ell\}$)

1: **Out**:= $\emptyset$.
2: Compute the set $\mathcal{B}(S)$ containing all line segments, bounding a triangle of the input, that contribute to the boundary of S (*i.e.*, that contain an edge of the boundary). Meanwhile, construct the planar subdivision $\mathcal{U}(S)$ induced by the triangles composing S.
3: Compute the convex hull $\mathcal{CH}(S)$ of S.
4: Construct the doubly connected edge list DCEL(S), induced by the planar subdivision defined by the lines through the segments of $\mathcal{B}(S)$, using $\mathcal{CH}(S)$ as a bounding box.
5: **while** there are any unvisited half-edges in DCEL($\mathcal{S}$) left **do**
6: Let e be an unvisited edge.
7: $\Sigma_x := 0$, $\Sigma_y := 0$, count := 0, Elist := $\emptyset$.
8: **while** e is unvisited **do**
9: Mark e with the label *visited*.
10: Elist := Elist $\cup$ $\{(e \rightarrow \text{origin}, e \rightarrow \text{next} \rightarrow \text{origin})\}$, $\Sigma_x := \Sigma_x + e \rightarrow \text{origin} \rightarrow x$, $\Sigma_y := \Sigma_y + e \rightarrow \text{origin} \rightarrow y$, count := count + 1.
11: $e := e \rightarrow \text{next}$.
12: **end while**
13: $\mathbf{x} := (\frac{\Sigma_x}{\text{count}}, \frac{\Sigma_y}{\text{count}})$.
14: **if** the point $\mathbf{a}$ in $\mathbf{x}$ belongs to a face of $\mathcal{U}(S)$ **then**
15: **for all** elements $(\mathbf{a}_s, \mathbf{a}_e)$ of Elist **do**
16: **Out** := **Out** $\cup\{T_{\mathbf{a}\mathbf{a}_s\mathbf{a}_e}\}$, where $T_{\mathbf{a}\mathbf{a}_s\mathbf{a}_e}$ is the (closed) triangle with corner points $\mathbf{a}$, $\mathbf{a}_s$ and $\mathbf{a}_e$.
17: **end for**
18: **end if**
19: **end while**
20: **return Out**.

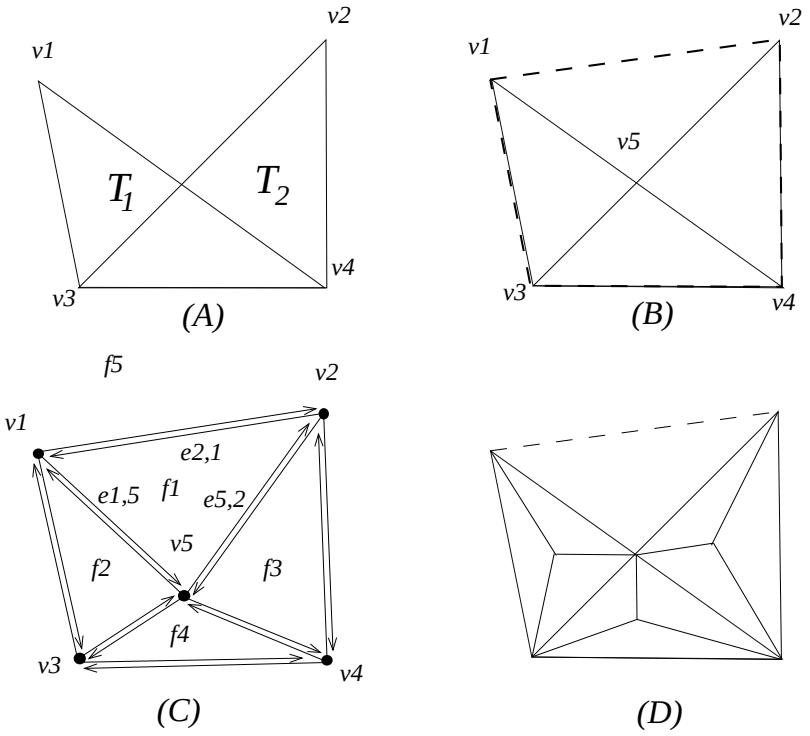


Figure 6.4: The different steps of Algorithm 3 applied to $S = \{T_1, T_2\}$. In this example, all boundary segments of all triangles of S contribute to the boundary of S. The line arrangement induced by the carriers of the edges of the input triangles is bounded by the convex hull of the input (Part (B)). A doubly-connected edge list is constructed out of the line arrangement, storing its topological structure (Part (C)). Finally, the triangulation is computed (Part (D)).

Property 6.2.1 (**Algorithm 3 describes a triangulation method**)**.** Let S be a snapshot. The output $\mathcal{T}_S(S)$ of Algorithm 3 applied to S is a triangulation of S.

Proof. Let $S = \{T_1, T_2, \ldots, T_k\}$ be a snapshot. It is easy to see that the output $\mathcal{T}_S(S) = \{T'_1, T'_2, \ldots, T'_\ell\}$ of $\mathcal{T}_S$ is a triangulation. By construction, $\mathcal{T}_S(S)$ is a set of triangles that either have no intersection, or share a corner point or bounding segment. It is clear from the algorithm that $\bigcup_{i=1^k} T_i = \bigcup_{j=1^\ell} T'_j$, because each triangle in $\mathcal{T}_S(S)$ is tested for membership of S. The other direction also holds, because initially, the convex hull of S is triangulated, which contains S.

Property 6.2.2 (**Quadratic output complexity**)**.** Given a snapshot $S = \{T_1, T_2, \ldots, T_m\}$, represented by m triangles. The triangulation $\mathcal{T}_S(S)$, where $\mathcal{T}_S$ is the triangulation method described in Algorithm 3, contains $O(m^2)$ triangles.

Proof. It is well-known (e.g., [16]), that an arrangement of m lines in the plane results in a subdivision of the plane containing $O(m^2)$ lines, $O(m^2)$ edges and $O(m^2)$

Half-edge	Origin	Twin	IncidentFace	Next	Prev
$e_{1,2}$	$v1$	$e_{2,1}$	$f5$	$e_{2,4}$	$e_{3,1}$
$e_{2,1}$	$v2$	$e_{1,2}$	$f1$	$e_{1,5}$	$e_{5,2}$
$e_{1,3}$	$v1$	$e_{3,1}$	$f2$	$e_{3,5}$	$e_{5,2}$
$e_{3,1}$	$v3$	$e_{1,3}$	$f5$	$e_{1,2}$	$e_{4,3}$
$e_{1,5}$	$v1$	$e_{5,1}$	$f1$	$e_{5,2}$	$e_{2,1}$
$e_{5,1}$	$v5$	$e_{1,5}$	$f2$	$e_{1,3}$	$e_{3,5}$
$e_{2,4}$	$v2$	$e_{4,2}$	$f5$	$e_{4,3}$	$e_{1,2}$
$e_{4,2}$	$v4$	$e_{2,4}$	$f3$	$e_{2,5}$	$e_{5,4}$
$e_{2,5}$	$v2$	$e_{5,2}$	$f3$	$e_{5,4}$	$e_{4,2}$
$e_{5,2}$	$v5$	$e_{2,5}$	$f1$	$e_{2,1}$	$e_{1,5}$
$e_{3,4}$	$v3$	$e_{4,3}$	$f4$	$e_{4,5}$	$e_{5,3}$
$e_{4,3}$	$v4$	$e_{3,4}$	$f5$	$e_{3,1}$	$e_{2,4}$
$e_{3,5}$	$v3$	$e_{5,3}$	$f2$	$e_{5,1}$	$e_{1,3}$
$e_{5,3}$	$v5$	$e_{3,5}$	$f4$	$e_{3,4}$	$e_{4,5}$
$e_{4,5}$	$v4$	$e_{5,4}$	$f4$	$e_{5,3}$	$e_{3,4}$
$e_{5,4}$	$v5$	$e_{4,5}$	$f3$	$e_{4,2}$	$e_{2,5}$

Table 6.1: The half-edge records of the doubly-connected edge list corresponding to Figure 6.4.

faces. It follows that the structure DCEL($\mathcal{S}$) will contain $O(m^2)$ half-edges, *i.e.*, two half-edges for each edge in the arrangement. In the worst case scenario, when all faces of the partition of the bounding box belong to S, one triangle is added to the output for each half-edge in DCEL($\mathcal{S}$) (connecting that half-edge with the center of mass of the face it bounds). Therefor, the output contains $O(m^2)$ triangles. □

In the following analysis of the running time of Algorithm 3, we assume that triangles are represented as triples of points, and that a point is represented as a pair of real numbers. We further assume that all basic arithmetic operations on coordinates of points require constant time.

Property 6.2.3 ($O(m^2 \log m)$ **running time**)**.** Given a snapshot $S = \{T_1, T_2, \ldots, T_m\}$ containing m triangles. The triangulation method $\mathcal{T}_S$, described in Algorithm 3, computes the triangulation $\mathcal{T}_S(S)$ of S in time $O(m^2 \log m)$.

Proof. Given a snapshot $S = \{T_1, T_2, \ldots, T_m\}$. Using a plane-sweep algorithm [19], we compute both the list of segments contributing to the boundary of S and the planar subdivision $\mathcal{U}(S)$ induced by $\bigcup_{i=1}^{m} T_i$. This takes $O(m^2 \log m)$, as there are at most m^2 intersection points between boundary segments of triangles of S.

The n triangles composing S together have at most $3m$ different corner points. Computing the convex hull of m points in the plane can be done in time $O(m \log m)$ (see [17]). The same authors propose, in [16], an algorithm to compute a doubly-connected edge list, representing an arrangements of m lines, in time $O(m^2)$. We next show that the changes we make to this algorithm do not influence its running time. So, as $\mathcal{B}(S)$ contains at most all $3m$ line segments, it induces an arrangement of at most $3m$ lines. Hence, Step 3 of Algorithm 3 also takes time $O(m^2)$.

Line(s)	Step	Time complexity
2	Compute $\mathcal{B}(S)$ and $\mathcal{U}(S)$	$O(m^2 \log m)$
3	Compute $\mathcal{CH}(S)$	$O(m \log m)$
4	Compute DCEL($\mathcal{S}$)	$O(m^2)$
$5-19$	Polygon extraction and triangulation	$O(m^2 \log m)$
	Overall time complexity	$O(m^2 \log m)$

Table 6.2: The time complexity of the various parts of Algorithm 3, when the input is a snapshot represented by n triangles.

We changed the original algorithm [16] for computing the doubly-connected edge list of an arrangement of lines as follows:

(i) We computed the *convex hull* of the input to serve as a bounding box instead of an axis-parallel rectangle containing all intersection points of the arrangement. The complexity of computing such an axis-parallel rectangle is higher ($O(m^2)$) than that of computing the convex hull ($O(m \log m)$).

(ii) The cost of constructing the doubly-connected edge list of the convex hull is $O(m)$, as the convex hull contains at most $3m$ corner points and the algorithm for computing it, as described in [17], already outputs the corner points of the convex hull in circular order. In the original algorithm [16] with an axis-parallel bounding rectangle, computing the doubly-connected edge list of this rectangle only takes constant time. This extra time does, however, not affect the overall complexity.

(iii) The next step of both algorithms involves finding the intersection points between the lines to be inserted and the partial arrangement induced by the previously inserted lines. In the original algorithm, this is easier only for the intersection of a line with the bounding box. For the intersections with all other lines in the arrangement, the cost is the same.

The next part of Algorithm 3 (starting from Line 5) takes time $O(m^2 \log m)$. Each half-edge of the doubly-connected edge list is visited only once. Also, each half-edge is only inserted once into the set *Elist*, and consulted only once therein to create a triangle. As an arrangement of m lines in the plane results in $O(m^2)$ edges, the number of half-edges in DCEL($\mathcal{S}$) also is $O(m^2)$. We can, in time $O(m^2)$, preprocess $\mathcal{U}(S)$ into a structure that allows point location in $O(\log m)$ time [22]. Therefor, testing for each of the $O(m^2)$ centers of mass whether they are part of the input takes $O(m^2 \log m)$.

We can conclude that all parts of Algorithm 3 run in time $O(m^2 \log m)$. □

Table 6.2 summarizes the computational complexity of the various parts of Algorithm 3.

Property 6.2.4 ($\mathcal{T}_S$ **is affine-invariant**)**.** The triangulation method $\mathcal{T}_S$ is affine-invariant.

Proof. According to the definition of affine-invariance of spatial triangulation methods (Definition 6.2), we have to prove the following. Let $A = \{T_{a,1}, T_{a,2}, \ldots, T_{a,k}\}$ and $B = \{T_{b,1}, T_{b,2}, \ldots, T_{b,\ell}\}$ be snapshots, given as sets of triangles, such that there exists an affinity $\alpha : \mathbb{R}^2 \to \mathbb{R}^2$ for which $B = \alpha(A)$. Then, for each triangle T of $\mathcal{T}_S(A)$, it holds that the triangle $\alpha(T)$ is a triangle of $\mathcal{T}_S(B)$.

We prove this by going through the steps of the triangulation procedure $\mathcal{T}_S$. Let $A = \{T_{a,1}, T_{a,2}, \ldots, T_{a,k}\}$ and $B = \{T_{b,1}, T_{b,2}, \ldots, T_{b,\ell}\}$ be such snapshots.

The convex hull and boundary of spatial figures are both affine-invariant (more specific, the boundary is a topological invariant). Intersection points between lines and the order of intersection points on one line with other lines are affine-invariant (even topological invariant). The subdivision of the convex hull $\mathcal{CH}(B)$ of B induced by the arrangement of lines through the boundary of B is hence the image under α of the subdivision of the convex hull $\mathcal{CH}(A)$ of A induced by the arrangement of lines through the boundary of A. The doubly-connected edge list only stores topological information about the arrangement of lines, *i.e.*, which edges are incident to which vertices and faces. Naturally, this information is preserved by affine transformations. The center of mass of a convex polygon is an affine invariant. Finally, the fact that a triangle is inside the boundary of the input and the fact that it is not are both affine-invariant. This completes the proof. □

Summarizing this section, we proposed a spatial triangulation method that, given a snapshot consisting of m triangles, returns an affine-invariant triangulation of this snapshot containing $O(m^2)$ triangles, in time $O(m^2 \log m)$.

We remark here that the idea of using carriers of boundary segments to partition figures was also used in an algorithm to decompose semi-linear sets by Dumortier, Gyssens, Vandeurzen and Van Gucht [21].

In the next section, we will use the affine triangulation method $\mathcal{T}_S$ to construct a spatio-temporal triangulation of geometric objects.

6.3 An Affine-invariant Spatio-temporal Triangulation Method

In this section, we present an spatio-temporal triangulation algorithm that takes as input a geometric object, *i.e.*, a finite set of atomic objects of the class $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$. We will adapt the spatial triangulation method $\mathcal{T}_S$, described in Algorithm 3, for time-dependent data.

The proposed spatio-temporal triangulation algorithm $\mathcal{T}_{ST}$ will have three main construction steps. First, in the *partitioning step*, the time domain of the geometric object will be partitioned into a set of points and open time intervals. For each element of this partition, all its snapshots have an *isomorphic triangulation*, when computed by the method $\mathcal{T}_S$. We refer to Definition 6.8 below for a formal definition of this isomorphism. Second, in the *triangulation step*, the spatio-temporal triangulation is computed for each element in the time partition, using the fact that all snapshots have *isomorphic triangulations*. Third, in the *merge step*, we merge objects when possible, to obtain a unique (and minimal) triangulation.

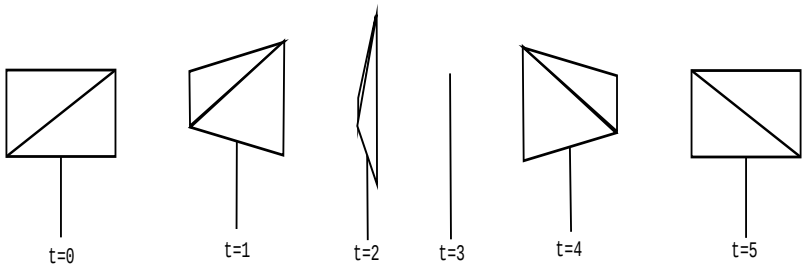


Figure 6.5: Snapshots of a traffic sign as seen by an observer circularly moving around it.

We will start this section by defining isomorphic triangulations. Then we explain the different steps of the algorithm for computing a spatio-temporal affine-invariant triangulation of geometric objects separately. We illustrate the algorithm with an example and end with some properties of the triangulation.

Definition 6.8 (**$\mathcal{T}_S$-isomorphic snapshots**)**.** Let S_1 and S_2 be two snapshots of a geometric object. We say that S_1 and S_2 are *$\mathcal{T}_S$-isomorphic*, denoted $S_1 \equiv_{\mathcal{T}_S} S_2$, if there exists a bijective mapping $h : \mathbb{R}^2 \to \mathbb{R}^2$ with the following property: A triangle $T = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ of $\mathcal{T}_S(S_1)$ is incident to the triangles $T_{1,2}$, $T_{2,3}$ and $T_{3,1}$ (where each $T_{i,((i+1) \mod 3)}$ is either a triangle of $\mathcal{T}_S(S_1)$ that shares the segment $\mathbf{a}_i\mathbf{a}_{((i+1) \mod 3)}$ with T, a triangle of $\mathcal{T}_S(\mathcal{CH}(S_1) \setminus S_1)$ that shares the segment $\mathbf{a}_i\mathbf{a}_{((i+1) \mod 3)}$ with T, or is ϵ, which means that no triangle shares that boundary segment with T) if and only if, the triangle $h(T) = (h(\mathbf{a}_1), h(\mathbf{a}_2), h(\mathbf{a}_3))$ belongs to $\mathcal{T}_S(S_2)$ and is bounded by $h(T_{1,2})$, $h(T_{2,3})$ and $h(T_{3,1})$. Moreover, if $T_{i,((i+1) \mod 3)}$ is a triangle of $\mathcal{T}_S(S_1)$, then $h(T_{i,((i+1) \mod 3)})$ is a triangle of $\mathcal{T}_S(S_2)$ that shares the line segment $h(\mathbf{a}_i)h(\mathbf{a}_{((i+1) \mod 3)})$ with $h(T)$, if $T_{i,((i+1) \mod 3)}$ is a triangle of $\mathcal{T}_S(\mathcal{CH}(S_1) \setminus S_1)$, then $h(T_{i,((i+1) \mod 3)})$ is a triangle of $\mathcal{T}_S(\mathcal{CH}(S_2) \setminus S_2)$ that shares the line segment $h(\mathbf{a}_i)h(\mathbf{a}_{((i+1) \mod 3)})$ with $h(T)$ and if $T_{i,((i+1) \mod 3)}$ equals ϵ, then so does $h(T_{i,((i+1) \mod 3)})$.

So, two snapshots S_1 and S_2 are $\mathcal{T}_S$-isomorphic if the triangles in $\mathcal{T}_S(S_1) \cup \mathcal{T}_S(\mathcal{CH}(S_1) \setminus S_1)$ and $\mathcal{T}_S(S_2) \cup \mathcal{T}_S(\mathcal{CH}(S_2) \setminus S_2)$ have the same (topological) adjacency graph. In particular, if S_1 and S_2 are equal up to an affinity of $\mathbb{R}^2$, then they are $\mathcal{T}_S$-isomorphic.

Example 6.9. The triangulations shown in Figure 6.1 are $\mathcal{T}_S$-isomorphic to each other. In Figure 6.5, all snapshots shown except the one at time moment $t = 3$ are $\mathcal{T}_S$-isomorphic. The snapshot at time moment $t = 3$ is clearly not isomorphic to the others, since it consists only of one line segment. □

Remark that for Figure 6.1, the mapping h is an affinity. In Figure 6.5, this is not the case.

Now, we introduce a spatio-temporal triangulation method $\mathcal{T}_{ST}$ that constructs a time-dependent affine triangulation of spatio-temporal objects that are represented

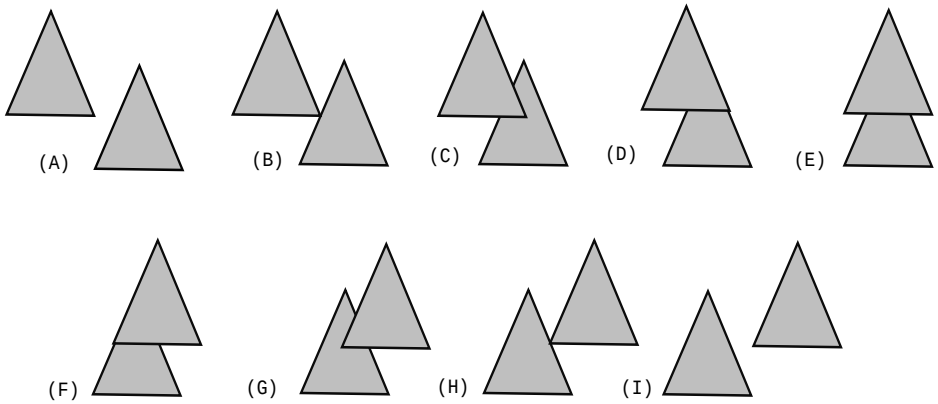


Figure 6.6: The snapshots at time moments $t = \frac{1}{4}$ (A), $t = \frac{1}{2}$ (B), $t = 1$ (C), $t = \frac{3}{2}$ (D), $t = 2$ (E), $t = \frac{5}{2}$ (F), $t = 3$ (G), $t = \frac{7}{2}$ (H) and $t = 4$ (I) of the geometric object of Example 6.10.

by geometric objects. We will explain its three main steps, *i.e.*, the partitioning step, the triangulation step and the merge step separately in the next subsections.

We will illustrate each step on the following example.

Example 6.10. Let $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2\}$ be a geometric object, where $\mathcal{O}_1$ is given as $(((-1, 0), (1, 0), (0, 2)), [0, 4], Id)$ and $\mathcal{O}_2$ is given as $(((-3, 1), (-1, 1), (-2, 3)), [0, 4], f)$ and f is the element of $\mathcal{F}^{\mathrm{Rat}}_{\mathrm{Aff}}$ mapping triples (x, y, t) to pairs $(x + t, y)$. Figure 6.6 shows the snapshots of $\mathcal{O}$ at time moments $t = \frac{1}{4}$ (A), $t = \frac{1}{2}$ (B), $t = 1$ (C), $t = \frac{3}{2}$ (D), $t = 2$ (E), $t = \frac{5}{2}$ (F), $t = 3$ (G), $t = \frac{7}{2}$ (H) and $t = 4$ (I). □

Let $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_m = (S_m, I_m, f_m)\}$ be a geometric object of the class $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}^{\mathrm{Rat}}_{\mathrm{Aff}} \rangle$. We assume that the S_i are given as triples of points (*i.e.*, pairs of real numbers), the I_i as structures containing two real numbers and two flags (indicating whether the interval is closed on the left or right side) and, finally, the f_i are given as vectors of integer coefficients, for $i = 1, \ldots, m$.

6.3.1 The Partitioning Step.

Let $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_m = (S_m, I_m, f_m)\}$ be a geometric object. In the first step of $\mathcal{T}_{ST}$, the time domain I of $\mathcal{O}$, *i.e.*, the convex closure $\bigcup_{i=1}^{m} I_i$ of the union of all the time domains $I_i (i = 1 \ldots m)$ is partitioned in such a way that, for each element of that partition, all its snapshots are $\mathcal{T}_S$-isomorphic.

Recall that, in Chapter 5, we defined the *finite time partition* $\mathcal{P}$ (see Definition 5.7) of the time domain of two atomic objects in such a way that for each element P of $\mathcal{P}$, the carrier sets of each snapshot of P are topologically equivalent. This definition can easily be extended to an arbitrary number of atomic objects. Also Property 5.2.5, stating that the finite time partition exists, still holds in the extended setting. To keep this chapter self-contained, we explicitly give the extended version of Definition 5.7 and Property 5.2.5:

Definition 6.11 (**Generalized finite time partition**). We call a *finite time partition* of a geometric object $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m\}$ any partition of the interval $I = \bigcup_{i=1}^{m} I_i$ into a finite number of time intervals $J_1, \ldots, J_k$ such that for any $\tau, \tau' \in J_\ell$ (and all $1 \leq \ell \leq k$), $\bigcup_{i=1}^{m} car(f_i(S_i, \tau))$ and $\bigcup_{i=1}^{m} car(f_i(S_i, \tau'))$ are topologically equivalent sets in $\mathbb{R}^2$.

Here, two subsets A and B of $\mathbb{R}^2$ are called *topologically equivalent* when there exists an orientation-preserving homeomorphism h of $\mathbb{R}^2$ such that $h(A) = B$.

Property 6.3.1 (**Existence of the generalized finite time partition**). Let $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m\}$ be a geometric object of the class $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$. There exists a finite time partition of $\mathcal{O}$.

The proof of this property is completely analog to the proof of Property 5.2.5 in Chapter 5.

We now proceed with the partitioning step of the spatio-temporal triangulation algorithm. In this step, a generalized finite time partition of $\mathcal{O}$ is computed, using the information of the *time-dependent* carriers of the atomic objects in $\mathcal{O}$. Each time an intersection point between two or more time-dependent carriers starts or ceases to exist, or when intersection points change order along a line, a new time interval of the partition is started. Given three *continuously moving* lines, the intersection points of the first line with the two other lines only change order along the first line, if there exists a moment where all three lines intersect in one point. Algorithm 4 describes the partitioning step in detail.

We will show later that the result of the generalized finite time partition is a set of intervals during which all snapshots are $\mathcal{T}_S$-isomorphic. This partition is, however, not the coarsest possible partition having this property, because there might be atomic objects that, during some time, are completely overlapped by other atomic objects. Therefor, we will later, after the triangulation step, again merge elements of the generalized finite time partition, whenever possible.

We illustrate Algorithm 4 on the geometric object of Example 6.10.

Example 6.12. Recall from Example 6.10 that $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2\}$, where $\mathcal{O}_1$ is given as $(((-1,0),(1,0),(0,2)),[0,4],Id)$ and $\mathcal{O}_2$ is given as $(((-3,1),(-1,1),(-2,3)),[0,4],f)$ and f is the element of $\mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}}$ mapping triples (x,y,t) to pairs $(x+t,y)$.

We now illustrate the partitioning algorithm on input $\mathcal{O}$. First, the list χ will contain the time moments 0 and 4. The list $\mathcal{C}$ will contain six elements. Table 6.3 shows these segments and the formulas describing their time-dependent carriers. All pairs of segments have an intersection that exists always, except for the pairs $(\mathcal{O}_{c,2}, \mathcal{O}_{c,5})$, $(\mathcal{O}_{c,3}, \mathcal{O}_{c,6})$ and $(\mathcal{O}_{c,1}, \mathcal{O}_{c,4})$. The intersections of $\mathcal{O}_{c,2}$ with $\mathcal{O}_{c,5}$ and $\mathcal{O}_{c,3}$ with $\mathcal{O}_{c,6}$ exist only at respectively $t = \frac{5}{2}$, $t = \frac{3}{2}$. The segments $\mathcal{O}_{c,1}$ and $\mathcal{O}_{c,4}$ never intersect. Of all possible triples of carriers, only two triples have a common intersection within the interval $[0,4]$. The carriers of $\mathcal{O}_{c,2}$, $\mathcal{O}_{c,4}$ and $\mathcal{O}_{c,6}$ intersect at $t = \frac{1}{2}$ and the carriers of $\mathcal{O}_{c,3}$, $\mathcal{O}_{c,4}$ and $\mathcal{O}_{c,5}$ intersect at $t = \frac{7}{2}$. The partitioning step will hence return the list

$$\chi = (0, \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, 4).$$

□

Algorithm 4 Partition (Input: $\mathcal{O} = \{\mathcal{O}_1, ..., \mathcal{O}_n\}$, Output: $\chi = \{\tau_1, \tau_2, \ldots, \tau_m\}$)

1: Let $\chi = (\tau_1 \leq \tau_2 \leq \ldots \leq \tau_k)(2 \leq k \leq 2n)$ be a sorted list of time moments that appear either as a begin or endpoint of I_i for any of the objects $\mathcal{O}_i = (S_i, I_i, f_i)$, $1 \leq i \leq n$.
2: $\mathcal{C} = \emptyset$.
3: **for all** atomic objects $\mathcal{O}_i = (S_i, I_i, f_i), 1 \leq i \leq n$ **do**
4: Add the new atomic objects $(S_{i,1}, I_i, f_i)$, $(S_{i,2}, I_i, f_i)$ and $(S_{i,3}, I_i, f_i)$ to $\mathcal{C}$, where $S_{i,1}$, $S_{i,2}$ and $S_{i,3}$ are the boundary segments of S_i.
5: **end for**
6: **for all** pairs of objects (S_{i,ℓ_1}, I_i, f_i) and (S_{j,ℓ_2}, I_j, f_j) of $\mathcal{C}$ $(1 \leq i < j \leq n; 1 \leq \ell_1, \ell_2 \leq 3)$ **do**
7: **if** $I_i \cap I_j \neq \emptyset$ **then**
8: Compute the end points of the intervals during which the intersection of the carriers of both time-dependent line segments does exist. Add those such end points that lie within the interval $I_i \cap I_j$ to χ, in a sorted way.
9: **end if**
10: **end for**
11: **for all** triples of objects (S_{i,ℓ_1}, I_i, f_i), (S_{j,ℓ_2}, I_j, f_j) and (S_{k,ℓ_3}, I_k, f_k) of $\mathcal{C}$ $(1 \leq i < j < k \leq n; 1 \leq \ell_1, \ell_2, \ell_3 \leq 3)$ **do**
12: **if** $I_i \cap I_j \cap I_k \neq \emptyset$ **then**
13: Compute the end points of the intervals during which the carriers of the three time-dependent line segments intersect in one point. Add those such end points that lie within the interval $I_i \cap I_j \cap I_k$ to χ, in a sorted way.
14: **end if**
15: **end for**
16: Return χ.

Element	Carrier
$\mathcal{O}_{c,1} = (((-1,0),(1,0)),[0,4],Id)$	$y = 0$
$\mathcal{O}_{c,2} = (((-1,0),(0,2)),[0,4],Id)$	$y = 2x + 2$
$\mathcal{O}_{c,3} = (((0,2),(1,0)),[0,4],Id)$	$y = -2x + 2$
$\mathcal{O}_{c,4} = (((-3,1),(-1,1)),[0,4],f)$	$y = 1$
$\mathcal{O}_{c,5} = (((-3,1),(-2,3)),[0,4],f)$	$y = 2x + 7 - 2t$
$\mathcal{O}_{c,6} = (((-2,3),(-1,1)),[0,4],f)$	$y = -2x - 1 + 2t$

Table 6.3: The elements of the list $\mathcal{C}$ during the execution of the partitioning algorithm (Algorithm 4) on the geometric object from Example 6.12.

We analyze both the output complexity and sequential time complexity of the partition step. First remark that the product of ℓ univariate polynomials of degree d is a polynomial of degree ℓd. Let the transformation function of an atomic object consists of rational coefficients, being fractions of polynomials of degree at most d. It follows that the time-dependent line segments and carriers can be defined using fractions of polynomials in t of degree $O(d)$. Also, the time-dependent intersection point of two such carriers and the time-dependent cross-ratio of an intersection point compared to two moving end points of a segment, can be defined using fractions of polynomials in t of degree $O(d)$.

Property 6.3.2 (**Partition output complexity**)**.** Given a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$ consisting of n atomic objects. Let d be the maximal degree of any polynomial in the definition of the transformation functions $f_i, 1 \leq i \leq n$. The procedure **Partition**, as described in Algorithm 4, returns a partition of $I = \bigcup_{i=1}^{n} I_i$ containing $O(n^3 d)$ elements.

Proof. It is clear that the list χ contains $O(n)$ elements after Line 1 of Algorithm 4. Indeed, at most two elements are added for each atomic object. The list $\mathcal{C}$ will contain at most $3n$ elements. For each atomic object with a reference object that is a "real" triangle, 3 elements will be added to $\mathcal{C}$. In the case that one or more corner points coincide, one or two objects will be added to $\mathcal{C}$.

Now we investigate the number of time moments that will be inserted to χ while executing the for-loop starting at Line 6 of Algorithm 4. The intervals during which the intersection of two time-dependent carriers exists are computed. The intersection of two time-dependent line segments doesn't exist at time moments where the denominator of the rational function defining it is zero. Because this denominator always is a polynomial P in t, it has at most $\deg(P)$ zeroes, where $\deg(P)$ denotes the degree of P. Accordingly, at most $\deg(P) = O(d)$ elements will be added to χ. Hence, in total, $O(n^2 d)$ time moments are added in this step.

For the intersections of three carriers, a similar reasoning can be used. Hence, during the execution of the for-loop starting at Line 11 of Algorithm 4, $O(n^3 d)$ elements are added to χ.

We can conclude that the list χ will contain $O(n^3 d)$ elements. □

Now we analyze the time complexity of **Partition**. We first point out that finding all roots of an univariate polynomial of degree d, with accuracy ϵ can be done in time $O(d^2 \log d \log\log(\frac{1}{\epsilon}))$ [57]. We will use the abbreviation $z(d, \epsilon)$ for the expression $O(d^2 \log d \log\log(\frac{1}{\epsilon}))$. Note also that, although the product of two polynomials of degree d is a polynomial of degree $2d$, the computation of the product takes time $O(d^2)$. To keep the proofs of the complexity results as readable as possible, we will consider the complexity of any manipulation on polynomials (computing zeros, adding or multiplying) to be $z(d, \epsilon)$, where a precision of ϵ is obtained.

Property 6.3.3 (**Partition computational complexity**)**.** Given a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$ consisting of n atomic objects. Let d be the maximal degree of any polynomial in the definition of the transformation functions $f_i, 1 \leq i \leq n$ and let ϵ be the desired precision for computing the zeros of polynomials. The procedure **Partition**, as described in Algorithm 4, returns a partition of $I = \bigcup_{i=1}^{n} I_i$ in time $O(n^3(z(d, \epsilon) + d \log n))$.

Proof. Let $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$ be a geometric object. Let d be the maximum degree of any of the polynomials used in the definition of the functions $f_i, 1 \leq i \leq n$.

Constructing the initial list χ, on Line 1, takes time $O(n \log n)$ (it is well known that the inherent complexity of sorting a list of n elements is $O(n \log n)$). Computing the set $\mathcal{C}$ can be done in time $O(nd)$: all n elements of $\mathcal{O}$ are considered, and the time needed to copy the transformation functions f_i depends on the maximal degree the polynomials defining them have. Recall that $\mathcal{C}$ contains at most $3n$ elements.

The first for-loop, starting at Line 6 of Algorithm 4 is executed $O(n^2)$ times. One execution of its body takes $z(d, \epsilon)$. Indeed, computing the formula representing the time-dependent intersection, checking whether its denominator is always zero and finding the zeros of the denominator (a polynomial of degree linear in d) have all time complexity $z(d, \epsilon)$. Therefor, the first for-loop takes time $O(n^2 z(d, \epsilon))$ in total.

The second for-loop has time complexity $O(n^3 z(d, \epsilon))$. The reasoning here is the same as for the previous for-loop.

Finally, sorting the list χ, which contains $O(n^3 d)$ elements at the end, requires time $O(n^3 d \log(nd))$.

If we summarize the complexity of all the separate steps, we obtain $O(n^3(z(d, \epsilon) + d \log n))$. □

We now proceed with the triangulation step.

6.3.2 The Triangulation Step.

Starting with a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$, the partitioning algorithm identifies a list χ of time moments that is used to partition the time domain $I = \bigcup_{i=1}^{n} I_i$ of $\mathcal{O}$ into points and open intervals. For each element in that partition (point or open interval), we now triangulate the part of $\mathcal{O}$ restricted to that point or open interval.

The triangulation of the snapshots of $\mathcal{O}$ at the time moments in χ is straightforward. For each of the time moments τ of χ, the spatial triangulation method $\mathcal{T}_S$ is applied to the snapshot $\mathcal{O}^\tau$. For each of the triangles T in $\mathcal{T}_S(\mathcal{O}^\tau)$, an atomic object is constructed with T as reference object, the singleton $\{\tau\}$ as time domain and the identity as its transformation function.

The triangulation of the parts of $\mathcal{O}$ restricted to the open intervals in the time partition requires a new technique. We can however benefit from the fact that throughout each interval, all snapshots of $\mathcal{O}$ have an $\mathcal{T}_S$-isomorphic triangulation. For each of the open intervals defined by two subsequent elements $]\tau_j, \tau_{(j+1)}[$ of χ, we compute the snapshot at the middle $\tau_m = \frac{1}{2}(\tau_j + \tau_{(j+1)})$ of $]\tau_j, \tau_{(j+1)}[$ and its triangulation $\mathcal{T}_S(\mathcal{O}^{\tau_m})$. Each triangle boundary segment that contributes to the boundary of $\mathcal{O}^{\tau_m}$ at time moment τ_m, will also contribute to the boundary of $\mathcal{O}$ at the snapshot of $\mathcal{O}$ at any time moment $\tau \in]\tau_j, \tau_{(j+1)}[$. So, the moving line segment can be considered a boundary segment throughout $]\tau_j, \tau_{(j+1)}[$. If two carriers of boundary segments intersect at time moment τ_m, the intersection of the moving segments will exist throughout $]\tau_j, \tau_{(j+1)}[$, and so on. Therefor, we will compute the spatial triangulation of the snapshot $\mathcal{O}^{\tau_m}$ using the procedure $\mathcal{T}_S$, but we will copy every action on a point or line segment at time moment τ_m on the moving point of line segment of which the point or

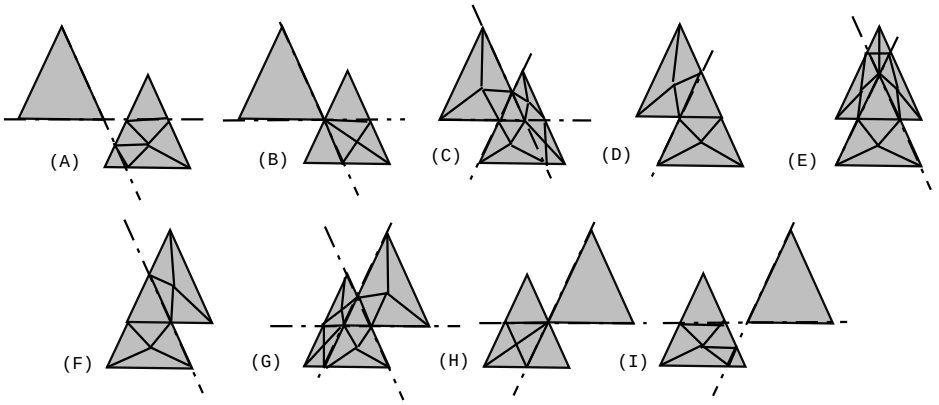


Figure 6.7: The triangulations of the objects of Example 6.10 at time moments $t = \frac{1}{4}$ (A), $t = \frac{1}{2}$ (B), $t = 1$ (C), $t = \frac{3}{2}$ (D), $t = 2$ (E), $t = \frac{5}{2}$ (F), $t = 3$ (G), $t = \frac{7}{2}$ (H) and $t = 4$ (I).

segment is a snapshot. The triangles returned by the spatial triangulation algorithm when applied to $\mathcal{O}^{\tau_m}$ will be reference objects for the atomic objects, returned by the spatio-temporal triangulation algorithm. These atomic objects exist during the interval $]\tau_j, \tau_{(j+1)}[$. Knowing the functions representing the time-dependent corner points of the triangles (because of the copying), together with the time interval and the reference object, we can deduce the transformation function and construct atomic objects (see Lemma 5.8 of Chapter 5 for the formula computing this transformation).

Next, a detailed description of the spatio-temporal triangulation is given in Algorithm 5. In this description of the spatio-temporal triangulation procedure, we will use the data type *Points* which is a structure containing a (2-dimensional) point (represented using a pair of real numbers), a pair of rational functions of t (a rational function is represented using a pair of vectors of integers, denoting the coefficients of a polynomial), representing a moving point, and finally a time interval (represented as a pair of real numbers and two flags indicating whether the interval is open or closed at each end point). We will only use or fill in this time information when mentioned explicitly. Given an element Pt of type *Points*, we address the point it stores by $Pt \rightarrow Point$, the functions of time by $Pt \rightarrow f_x$ and $Pt \rightarrow f_y$ respectively, and the begin and end point of the time interval by $Pt \rightarrow I_b$ and $Pt \rightarrow I_e$. The flags $Pt \rightarrow C_b$ and $Pt \rightarrow C_e$ are true when the interval is closed at its begin or end point respectively. A pair of elements of the type *Points* is denoted an element of the type *Segments*.

We again illustrate the spatio-temporal triangulation algorithm on the geometric object of example 6.10.

Example 6.13. Recall from Example 6.10 that $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2\}$, where $\mathcal{O}_1$ is given as $(((-1,0),(1,0),(0,2)),[0,4],Id)$ and $\mathcal{O}_2$ is given as $(((-3,1),(-1,1),(-2,3)),[0,4],f)$ and f is the element of $\mathcal{F}^{\mathrm{Rat}}_{\mathrm{Aff}}$ mapping triples (x,y,t) to pairs $(x+t,y)$.

From Example 6.12, we recall that the output of the procedure **partition** on input $\mathcal{O}$ was the list $\chi = (0, \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, 4)$.

Algorithm 5 Triangulate (Input: $\mathcal{O} = \{\mathcal{O}_1, ..., \mathcal{O}_n\}, \chi = \{\tau_1, \ldots, \tau_k\}$, Output = $\{\mathcal{O}'_1, \ldots, \mathcal{O}'_\ell\}$)

for all time moments $\tau_j, j = 1 \ldots k$, of χ **do**
  for all triangles T in $\mathcal{T}_S(\mathcal{O}^{\tau_j})$ **do**
    return the atomic element $(T, \{\tau_j\}, Id)$.
  end for
end for
Let $S_<$ be the list containing all atomic objects $\mathcal{O}_i = (S_i, I_i, f_i), 1 \leq i \leq n$, sorted by the begin points $I_{i,b}$ of their time domains.
Let S_{Active} be a list of elements of the type *Segments*, $S_{\text{Active}} = ()$.
for all pairs (τ_j, τ_{j+1}), $j = 1 \ldots (k-1))$, in χ **do**
  τ_m:= $\frac{1}{2}(\tau_j + \tau_{j+1})$.
  Remove all elements (Pt_1, Pt_2) of S_{Active} for which $\tau_j = Pt_1 \rightarrow I_e = Pt_2 \rightarrow I_e$.
  for all elements (Pt_1, Pt_2) remaining in S_{Active} **do**
    $Pt_r \rightarrow Point := (Pt_r \rightarrow f_x(\tau_m), Pt_r \rightarrow f_y(\tau_m))$, $r = 1, 2$.
  end for
  for all $\mathcal{O}_i = (S_i = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3), I_i, f_i)$ in $S_<$ for which $I_{i,b}$ is τ_i **do**
    Construct three *Points* Pt_1, Pt_2 and Pt_3 such that $Pt_r \rightarrow Point = \mathbf{a}_r$, $Pt_r \rightarrow f_x = f_i(a_{r,x}, \tau_m)$, $Pt_r \rightarrow f_y = f_i(a_{r,y}, \tau_m)$ and $Pt_r \rightarrow I_b$ and $Pt_r \rightarrow I_e$ respectively contain τ_j and τ_{j+1} $(r = 1, \ldots, 3)$.
    Construct three *Segments* St_1, St_2 and St_3, containing two different elements from the set $\{Pt_1, Pt_2, Pt_3\}$. Add them to S_{Active}.
  end for
  Compute the set $\mathcal{B}^t(S_{\text{Active}})$ of elements of the type *Segments*, using only the constant point information of the elements of S_{Active}. Meanwhile, construct the subdivision $\mathcal{U}(\mathcal{O}^{\tau_m})$.
  Compute the convex hull $\mathcal{CH}^t(S_{\text{Active}})$, using only the constant point information of the elements of S_{Active}, a list of elements of the type *Points*.
  Construct DCEL$^t(S_{\text{Active}})$, where each half-edge (resp. origin) is now an element of the type *Segments* (resp. *Points*). Use $\mathcal{CH}^t(S_{\text{Active}})$ as a bounding box. Each time the intersection of two constant carriers is computed, also compute the formula representing the moving intersection point.
  while there are any unvisited *Segments* St in DCEL$^t(\mathcal{S})$ left **do**
    Compute the list E^{t}list of *Segments* that form a convex polygon. Compute the *Points* structure Pt_m containing both the constant and time-dependent center of mass of that polygon
    if $Pt_m \rightarrow Point$ belongs to a face of $U(\mathcal{O}^{\tau_m})$ **then**
      for all elements $St = (Pt_1, Pt_2)$ of E^{t}list **do**
        Output the atomic object (S, I, f), where S is the triangle with corner points $Pt_1 \rightarrow Point$, $Pt_2 \rightarrow Point$ and $Pt_m \rightarrow Point$ and I is $]\tau_j, \tau_{j+1}[$. The transformation function f is computed using the functions $Pt_1 \rightarrow f_x$, $Pt_1 \rightarrow f_y$, $Pt_2 \rightarrow f_x$, $Pt_2 \rightarrow f_y$, $Pt_m \rightarrow f_x$ and $Pt_m \rightarrow f_y$.
      end for
    end if
  end while
end for

The triangulation of the snapshots at one of the time moments in χ are shown in Figure 6.7. To keep the example as simple as possible, we did not further triangulate convex polygons that are triangles already.

The open intervals to be considered are $]0, \frac{1}{2}[$, $]\frac{1}{2}, \frac{3}{2}[$, $]\frac{3}{2}, \frac{5}{2}[$, $]\frac{5}{2}, \frac{7}{2}[$ and $]\frac{7}{2}, 4[$. We illustrate the triangulation of the interval $]0, \frac{1}{2}[$. During the time interval $]0, \frac{1}{2}[$, the triangulation will always look like the one shown in Part (A) of Figure 6.7. Hence, $\mathcal{O}_2$ will not change, and $\mathcal{O}_1$ will be partitioned into seven triangles. The top one will not change, so the atomic object $(((0,2),(1,\frac{-1}{2}),(1,\frac{1}{2})),]0,\frac{1}{2}[,Id)$ will be part of the output. For the others, we have to compute the time-dependent intersections between the carriers and afterwards apply the formula from Lemma 5.8 of Chapter 5. We illustrate this for $\mathcal{O}_2$. the snapshot of $\mathcal{O}_2$ at the middle point $\frac{1}{4}$ of $]0, \frac{1}{2}[$ is the triangle with corner points $(\frac{-11}{4}, 1)$, $(\frac{-3}{4}, 1)$ and $(\frac{-7}{4}, 3)$. Its time-dependent corner points are $(-3+t, 1)$, $(-1+t, 1)$ and $(-2+t, 3)$. Solving the matrix equation

$$\begin{pmatrix} \frac{-11}{4} & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-11}{4} & 1 & 0 & 1 \\ \frac{-7}{4} & 3 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-7}{4} & 3 & 0 & 1 \\ \frac{-3}{4} & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-3}{4} & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} a(t) \\ b(t) \\ c(t) \\ d(t) \\ e(t) \\ f(t) \end{pmatrix} = \begin{pmatrix} -3+t \\ 1 \\ -2+t \\ 3 \\ -1+t \\ 1 \end{pmatrix}$$

gives the transformation function f' that maps triples (x, y, t) to pairs $(x - \frac{1}{4} + t, y)$. □

We also give the output complexity and time complexity for this triangulation step.

Property 6.3.4 (**Triangulation step: output complexity**)**.** Given a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$ consisting of n atomic objects and a finite partition χ of its time domain into k time points and $k-1$ open intervals. The procedure **Triangulation**, as described in Algorithm 5, returns $O(n^2k)$ atomic objects.

Proof. The number of atomic objects returned by the triangulation procedure for one time interval is the same as the number of triangles returned by the spatial triangulation method on a snapshot in that interval. We know from Property 6.2.2 that the number of triangles in the triangulation of a snapshot composed from n triangles is $O(n^2)$. Since their are $O(k)$ moments and intervals for which we have to consider such a triangulation, or a slightly adapted version of it, this gives $O(n^2k)$. □

Property 6.3.5 (**Triangulation step: computational complexity**)**.** Given a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$ consisting of n atomic objects and a finite partition χ of its time domain into k time points and $k-1$ open intervals. Let d be the maximal degree of any polynomial in the definition of the transformation functions $f_i, 1 \leq i \leq n$ and let ϵ be the desired precision for computing the zeros of polynomials. The procedure **Partition**, as described in Algorithm 4, returns a spatio-temporal triangulation of $\mathcal{O}$ in time $O(kz(d,\epsilon)n^2 \log n)$.

Proof. The first for-loop of Algorithm 5 is executed k times. The time needed for computing the snapshot of one atomic object at a certain time moment is $z(d, \epsilon)$. The spatial triangulation algorithm $\mathcal{T}_S$ runs in time $O(n^2 \log n)$, as was shown in Property 6.2.3. So we can conclude that the body of the first for-loop needs $O(n^2 \log n + nz(d, \epsilon))$ time. Sorting the atomic objects by their time domains takes $O(n \log n)$.

The second for loop is executed once for each open interval, defined by two consecutive elements of χ. In the body of this loop, first the list S_{Active} is updated. Each insertion or update takes time $z(d, \epsilon)$. At most all objects are in the list S_{Active}, so this part, described in the Lines 10 through 18 of Algorithm 5, needs time $O(nz(d, \epsilon))$. The next part, described in the Lines 19 through 29 essentially is the spatial triangulation algorithm, but, any time the intersection between two line segments is computed, also the rational functions defining the time-dependent intersection of their associated time-dependent line segments are computed. Computing those functions takes time $z(d, \epsilon)$. So the second part of the body of the second for loop requires $O(z(d, \epsilon)n^2 \log n)$.

If we add up the time complexity of two for-loops and the sorting step, we have $O(k(nz(d, \epsilon) + n^2 \log n) + n \log n + kz(d, \epsilon)(n + n^2 \log n))$, which is $O(kz(d, \epsilon)n^2 \log n)$. □

6.3.3 The Merge Step.

We already mentioned briefly in the description of the partitioning step that the partition of the time domain, as computed by Algorithm 4, might be finer than necessary. The partitioning algorithm takes into account all line segments, also those of objects that, during some time span, are entirely overlapped by other objects. To solve this, we merge as much elements of the time partition as possible.

The partition of the time domain is such that the merging algorithm will either try to merge a time point τ and an interval of the type $]\tau, \tau'[$ or $(\tau', \tau[$, or two different intervals of the type $(\tau'', \tau]$ and $]\tau, \tau'[$ or $(\tau'', \tau[$ and $[\tau, \tau'[$. Here, (and) can be either [or].

The simplest case is when a time moment and an interval have to be tested. Assume that these are $(\tau', \tau[$ and τ, respectively. These elements can be merged if there is a one to one mapping M from the atomic elements with time domain $(\tau', \tau[$ to those with time domain $\{\tau\}$ in the triangulation. Furthermore, for each pair of atomic objects $\mathcal{O}_1 = (S_1, (\tau', \tau[, f_1)$ and $\mathcal{O}_2 = (S_2, \{\tau\}, Id)$, $\mathcal{O}_2 = M(\mathcal{O}_1)$ if and only if the left limit $\lim_{t \to \tau} f_1(S_1, t) = S_2$. Note that, for rational functions f of t, $\lim_{t \to \tau} f(t)$ equals $f(\tau)$, provided that τ is in the domain of f [3].

When two intervals are to be merged, the procedure involves some more tests. Let $(\tau'', \tau[$ and $[\tau, \tau'[$ be the intervals to be tested. First, we have to verify that for each atomic object $\mathcal{O}_1 = (S_1, (\tau'', \tau[, f_1)$, $[\tau, \tau'[$ is in the domain of f_1 and that for each atomic object $\mathcal{O}_2 = (S_2, [\tau, \tau'[, f_2)$, $(\tau'', \tau[$ is in the domain of f_2. Second, we have to test whether $(\tau'', \tau[$ can be continuously expanded to $(\tau'', \tau]$. This involves the same tests as for the simple case where an interval and a point are tested. Finally, two atomic object can only be merged if the combined atomic object again is an

[3] Note that τ is in the domain of f only if all coefficients of the transformation function f are well-defined for $t = \tau$ and if the determinant of f is nonzero for $t = \tau$.

element of the class $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$. This means that, if S_2 would have been chosen as a reference object for $\mathcal{O}_1$, then f_1 would be equal to f_2, and vice versa. Using Lemma 5.8 of Chapter 5, this can be tested.

This merge step guarantees that the atomic objects exist maximally and that the resulting triangulation is the same for geometric objects that represent the same spatio-temporal object. Algorithm 6 shows this merging step in detail.

We illustrate Algorithm 6 on the geometric object of Example 6.10.

Example 6.14. Recall from Example 6.10 that $\mathcal{O} = \{\mathcal{O}_1, \mathcal{O}_2\}$, where $\mathcal{O}_1$ is given as $(((-1,0),(1,0),(0,2)),[0,4],Id)$ and $\mathcal{O}_2$ is given as $(((-3,1),(-1,1),(-2,3)),[0,4],f)$ and f is the element of $\mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}}$ mapping triples (x,y,t) to pairs $(x+t,y)$.

From Example 6.12, we recall that the output of the procedure **partition** on input $\mathcal{O}$ was the list $\chi = (0, \frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}, 4)$. This resulted in a partition of the interval $[0,4]$ consisting of the elements $\{0\}$, $]0, \frac{1}{2}[$, $\{\frac{1}{2}\}$, $]\frac{1}{2}, \frac{3}{2}[$, $\{\frac{3}{2}\}$, $]\frac{3}{2}, \frac{5}{2}[$, $\{\frac{5}{2}\}$, $]\frac{5}{2}, \frac{7}{2}[$, $\{\frac{7}{2}\}$, $]\frac{7}{2}, 4[$ and $\{4\}$. For each of these elements, (a snapshot of) their triangulation is shown in Figure 6.7.

During the merge step, the elements $t = 0$ and $]0, \frac{1}{2}[$ of the time partition will be merged. □

It is straightforward that the output and input of the merging algorithm have the same order of magnitude. Indeed, it is possible that no intervals are merged, and hence no objects. We discuss the computational complexity of the algorithm next. Note that the complexity is expressed in terms of the size of the input to the merging algorithm, which is the output of the spatio-temporal triangulation step.

Property 6.3.6 (**Merge step computational complexity**)**.** Given a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$, which is the output of the triangulation step, and a finite partition χ of its time domain into K time moments and open intervals. Let d be the maximal degree of any polynomial in the definition of the transformation functions $f_i (1 \leq i \leq n)$ and let ϵ be the desired precision for computing the zeros of polynomials. The procedure **Merge**, as described in Algorithm 6, merges the atomic objects in $\mathcal{O}$ in time $O(n \log \frac{n}{K} + nz(d, \epsilon))$.

Proof. Sorting all atomic objects by their time domains can be done in time $O(n \log n)$. Computing the list χ' can be straightforwardly done in time $O(K)$. This list will contain $2K - 1$ elements. We assume that $K > 1$ (in case $K = 1$ the merging algorithm is not applied). The while-loop starting at Line 4 of Algorithm 6, is executed at most $2K - 2$ times. Indeed, at each execution of the body of the while-loop, one new element of χ' is considered. The **if-else** structure in the body of the while-loop distinguishes three cases. All cases have the same time complexity, as they are analogous. We explain the first case in detail.

The number of atomic objects having the same time domain is of the order of magnitude of $O(\frac{n}{K})$. This follows from Property 6.3.4. The preprocessing of the snapshot takes $O(\frac{n}{K})$ time [22]. The for-loop, starting at Line 9 of Algorithm 6 is executed at most $O(\frac{n}{K})$ times. The time needed for checking whether an atomic object exists at some time moment and computing the snapshot (a triangle) is $z(d, \epsilon)$. Because of the preprocessing on the snapshot at time moment J_1, testing the barycenter of the triangle against that snapshot can be done in $O(\log \frac{n}{K})$ time [22]. In case the

Algorithm 6 Merge (Input: $\mathcal{O} = \{\mathcal{O}_1, ..., \mathcal{O}_n\}, \chi = \{\tau_1, \tau_2, \ldots, \tau_k\}$, Output: $\{\mathcal{O}'_1, \mathcal{O}'_2, \ldots, \mathcal{O}'_\ell\}$

Sort all atomic objects $\mathcal{O}_i$ by their time domains.
Let χ' be the list $(\tau_1,]\tau_1, \tau_2[, \tau_2, \ldots,]\tau_{k-1}, \tau_k[, \tau_k)$.
Let J_1 be the first element of χ' and J_2 the second.
while there are any elements in χ' left **do**
  $\mathcal{S}_1$ (resp. $\mathcal{S}_2$) is the set of all objects having J_1 (resp. J_2) as their time domain.
  if J_1 is a point **then**
    Preprocess the reference objects of the elements of $\mathcal{S}_1$ such that we can search the planar subdivision $\mathcal{U}_1$ they define.
    let **Found** be true.
    for all objects $\mathcal{O}_i = (S_i, J_2, f_i)$ in $\mathcal{S}_2$ **do**
      Check whether J_1 is part of the time domain of f_i.
      Compute their snapshot at time J_1 (which is a triangle T).
      Do a point location query with the center of mass of T in $\mathcal{U}_1$ and check whether the triangle found in $\mathcal{S}_1$ has the same coordinates as T. If not, **Found** becomes false.
      if **Found** is false **then**
        break;
      end if
    end for
    if **found** is true **then**
      remove all elements of $\mathcal{S}_1$ from $\mathcal{O}$ and extend the time domain of all elements of $\mathcal{S}_2$ to $J_1 \cup J_2$.
      $J_1 = J_1 \cup J_2$ and J_2 is the next element of χ' if any exists.
    else
      $J_1 = J_2$ and J_2 is the next element of χ', if any exists.
    end if
  else
    if J_2 is a point **then**
      do the same as in the previous case, but switch the roles of J_1 and J_2.
    else
      Let J'_1 be the element of $\{J_1, J_2\}$ the form $(\tau'', \tau[$ and J'_2 the one of the form $[\tau, \tau')$.
      Check whether $(\tau'', \tau[$ and $\{\tau\}$ can be merged.
      if this can be done **then**
        Check for each pair of matching atomic objects whether their transformation functions are the same (using Lemma 5.8 of Chapter 5).
      end if
    end if
  end if
end while

Step	Time complexity	Output complexity
Partition	$O(z(d,\epsilon)n^3 \log n)$	$O(n^3 d)$
Triangulate	$O(z(d,\epsilon)dn^5 \log n)$	$O(n^5 d)$
Merge	$O(n^5 d(\log n + z(d,\epsilon)))$	-
Overall	$O(z(d,\epsilon)dn^5 \log n)$	-

Table 6.4: The output and time complexity of the various parts of Algorithm 7, when the input is a geometric object of the class $\langle \mathcal{S}_{\mathrm{Tr}}, \mathcal{F}_{\mathrm{Aff}}^{\mathrm{Rat}} \rangle$, composed of n atomic objects, where the maximal degree of the polynomials describing the transformation functions is d and the desired precision for computing the zeros of polynomials is ϵ.

snapshots are the same, adjusting the time domains of all atomic objects takes time $O(\frac{n}{K})$. Summarizing, the time complexity of the first case is $O(\frac{n}{K} \log \frac{n}{K} + \frac{n}{K} z(d,\epsilon))$.

Combining this with the fact that the while-loop is executed $O(K)$ times, and the time complexity of the first two steps of the algorithm, we get an overall time complexity of $O(n \log \frac{n}{K} + nz(d,\epsilon))$. □

Finally, the spatio-temporal triangulation procedure $\mathcal{T}_{ST}$ combines the partition, triangulation and merging step. Algorithm 7 combines all steps.

Algorithm 7 $\mathcal{T}_{ST}$ (Input: $\mathcal{O} = \{\mathcal{O}_1, ..., \mathcal{O}_n\}$, Output: $\{\mathcal{O}'_1, \mathcal{O}'_2, \ldots, \mathcal{O}'_\ell\}$)

1: χ = **Partition**(O);
2: $\{\mathcal{O}''_1, \mathcal{O}''_2, \ldots, \mathcal{O}''_m\}$ = **Triangulate**($\mathcal{O}$, χ);
3: **if** χ has more than one element **then**
4: $\quad \{\mathcal{O}'_1, \mathcal{O}'_2, \ldots, \mathcal{O}'_\ell\}$ = **Merge**($\{\mathcal{O}''_1, \mathcal{O}''_2, \ldots, \mathcal{O}''_m\}$, χ);
5: $\quad$ return $\{\mathcal{O}'_1, \mathcal{O}'_2, \ldots, \mathcal{O}'_\ell\}$.
6: **else**
7: $\quad$ return $\{\mathcal{O}''_1, \mathcal{O}''_2, \ldots, \mathcal{O}''_m\}$.
8: **end if**

The following property follows from Property 6.3.2 and Property 6.3.4.

Property 6.3.7 ($\mathcal{T}_{ST}$ **output complexity**)**.** Given a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$ consisting of n atomic objects. Let d be the maximal degree of any polynomial in the definition of the transformation functions $f_i (1 \leq i \leq n)$. The spatio-temporal triangulation method $\mathcal{T}_{ST}$, as described in Algorithm 7, returns $O(n^5 d)$ atomic objects.

The next property follows from Property 6.3.3, Property 6.3.5 and Property 6.3.6. Table 6.4 summarizes the time complexity of the different steps.

Property 6.3.8 ($\mathcal{T}_{ST}$ **computational complexity**)**.** Given a geometric object $\mathcal{O} = \{\mathcal{O}_1 = (S_1, I_1, f_1), \mathcal{O}_2 = (S_2, I_2, f_2), \ldots, \mathcal{O}_n = (S_n, I_n, f_n)\}$ consisting of n atomic objects. Let d be the maximal degree of any polynomial in the definition of the transformation functions $f_i (1 \leq i \leq n)$ and let ϵ be the desired precision for

computing the zeros of polynomials. The spatio-temporal triangulation method $\mathcal{T}_{ST}$, as described in Algorithm 7, returns a spatio-temporal triangulation of $\mathcal{O}$ in time $O(z(d,\epsilon)dn^5 \log n)$.

We now show that Algorithm 7 describes an affine-invariant spatio-temporal triangulation method. We remark first that the result of the procedure $\mathcal{T}_{ST}$ is a spatio-temporal triangulation. Given a geometric object $\mathcal{O}$. It is clear that each snapshot of $\mathcal{T}_{ST}(\mathcal{O})$ is a spatial triangulation. Also, $st(\mathcal{O}) = st(\mathcal{T}_{ST}(\mathcal{O}))$. This follows from the fact that the time partition covers the whole time domain of $\mathcal{O}$ and that the method $\mathcal{T}_S$ produces a spatial triangulation.

Property 6.3.9 ($\mathcal{T}_{ST}$ **is affine-invariant**)**.** The spatio-temporal triangulation method $\mathcal{T}_{ST}$, described in Algorithm 7, is affine-invariant.

Proof. Recall Definition 6.2. *A spatio-temporal triangulation method* $\mathcal{T}_{ST}$ *is called* affine invariant *if and only if for any geometric objects* $\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ *and* $\{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\}$ *for which for each moment* τ_0 *in their time domains, there is an affinity* $\alpha_{\tau_0} : \mathbb{R}^2 \to \mathbb{R}^2$ *such that if* $\alpha_{\tau_0}(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}^{\tau_0}) = \{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\}^{\tau_0}$*, also* $\alpha_{\tau_0}(\mathcal{T}_{ST}(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\})^{\tau_0})$ *equals* $\mathcal{T}_{ST}(\{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\})^{\tau_0}$.

Let $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ and $\mathcal{O}' = \{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\}$ be geometric objects for which for each moment τ_0 in their time domains, there is an affinity $\alpha_{\tau_0} : \mathbb{R}^2 \to \mathbb{R}^2$ such that $\alpha_{\tau_0}(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}^{\tau_0}) = \{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\}^{\tau_0}$.

It follows from the construction of the spatio-temporal triangulation that $\mathcal{T}_{ST}(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\})^{\tau_0}$ equals $\mathcal{T}_S(\{\mathcal{O}_1, \ldots, \mathcal{O}_n\}^{\tau_0})$ and $\mathcal{T}_{ST}(\{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\})^{\tau_0}$ equals $\mathcal{T}_S(\{\mathcal{O}'_1, \ldots, \mathcal{O}'_n\}^{\tau_0})$. The property now follows from the affine-invariance of the spatial triangulation method $\mathcal{T}_S$. □

The following corollary follows straightforwardly from Property 6.3.9:

Corollary 6.15. Let $\mathcal{O} = \{\mathcal{O}_1, \ldots, \mathcal{O}_n\}$ and $\mathcal{O}' = \{\mathcal{O}'_1, \ldots, \mathcal{O}'_m\}$ be two geometric objects such that there is an affinity $\alpha : \mathbb{R}^2 \to \mathbb{R}^2$ such that, for each moment τ_0 in their time domains $\alpha(\mathcal{O}^{\tau_0}) = \mathcal{O}'^{\tau_0}$ holds. Then, for each atomic element (S, I, f) of $\mathcal{T}_{ST}(\mathcal{O})$, the element $(\alpha(S), I, f)$ belongs to $\mathcal{T}_{ST}(\mathcal{O}')$.

This shows that the partition is independent of the coordinate system used to represent the spatio-temporal object. The affine partitions of two spatio-temporal objects that are affine images of each other only differ in the coordinates of the spatial reference objects of the atomic objects.

7

Triangle-based Spatio-temporal Query Languages

In this chapter, we study affine-generic first-order logic queries, applied to databases consisting of (spatio-temporal) triangles, instead of n-dimensional points (as in Chapter 4) or real numbers (as in Chapter 3). When we were developing spatio-temporal point languages, in Chapter 4, we could build on the previous work of Gyssens, Van den Bussche and Van Gucht [42, 43] on spatial databases. Spatial triangle-based languages were not previously studied. After giving some definitions, we develop a spatial triangle logic and then extend our results to the case of spatio-temporal triangles (*i.e.*, triples of co-temporal points in $(\mathbb{R}^2 \times \mathbb{R})$.)

7.1 Notations

In this chapter, we introduce triangle variables and constants. We work in $\mathbb{R}^2$. Spatial triangle variables will be denoted $\triangle_1, \triangle_2, \ldots$. Constants containing such triples of points will be denoted $T_1, T_2, \ldots$, or $T_{\mathbf{abc}}$ when we want to emphasize the relationship between a triangle and its corner points a, b, $c \in \mathbb{R}^2$. We remark that triangles can be modelled as triples of points in $\mathbb{R}^2$. Occasionally, we will need to refer to the area of a triangle. As in the previous chapter, the area of a triangle T will be abbreviated $A(T)$.

We also introduce spatio-temporal triangles, which can be modelled as triples of moving points in $\mathbb{R}^2$. Variables referring to spatio-temporal triangles are distinguished from spatial triangle variables by a superscript: $\triangle_1^{st}, \triangle_2^{st}, \ldots$. The same holds for

constants, which are denoted $T_1^{st}, T_2^{st}, T_{pqr}^{st} \ldots$. We will also define triangle databases. For the spatial and spatio-temporal case respectively, we will use the symbols $\mathcal{D}$ and $\mathcal{D}^{st}$ to indicate triangle database instances.

The names of (spatio-temporal) triangle relations and database schemas containing such relation names will be recognizable by their hat: $\hat{R}$, $\hat{\sigma}$ and $\hat{R}^{st}$, $\hat{\sigma}^{st}$, respectively. Spatial and spatio-temporal point relation names and schemas are denoted $\dot{R}$ and $\dot{\sigma}$, $\dot{R}^{st}$ and $\dot{\sigma}^{st}$, respectively.

7.2 Definitions

We start with the definition of a *triangle database*, *i.e.*, a database that contains a (possibly infinite) collection of triangles. We define both spatial triangle databases and spatio-temporal triangle databases. We model triangles by triples of points of $\mathbb{R}^2$, *i.e.*, by elements of $(\mathbb{R}^2)^3$. Moving or changing (*i.e.*, spatio-temporal) triangles are modelled by sets of triples of co-temporal points in $(\mathbb{R}^2 \times \mathbb{R})$,*i.e.*, by sets of elements of $(\mathbb{R}^2 \times \{\tau_0\})^3$, for some $\tau_0 \in \mathbb{R}$. Triangles can degenerate, *i.e.*, corner points are allowed to coincide. For the remainder of this chapter, the term triangle refers to a triple of points. We refer to the set of points that is represented by a triangle as the *drawing* of that triangle.

Definition 7.1 (**Drawing of a triangle**). • Let $\triangle = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) \in (\mathbb{R}^2)^3$ be a spatial triangle. The *drawing* of $\triangle$ is the subset of $\mathbb{R}^2$ that is the convex closure of the points $\mathbf{a}_1$, $\mathbf{a}_2$ and $\mathbf{a}_3$.
• Let $\triangle^{st} = (p_1, p_2, p_3) \in (\mathbb{R}^2 \times \mathbb{R})^3$ be a spatio-temporal triangle. The *drawing* of $\triangle^{st}$ is the subset of co-temporal points of $(\mathbb{R}^2 \times \mathbb{R})$ that is the convex closure of the points p_1, p_2 and p_3.

In Chapter 3, the canonical bijection $can_{ST} : (\mathbb{R}^n \times \mathbb{R})^k \to \mathbb{R}^{(n+1)\times k}$ was introduced , mapping k-tuples of $(n+1)$-dimensional vectors to $((n+1) \times k)$-dimensional vectors of real numbers. We also define the bijection $can : (\mathbb{R}^n)^k \to \mathbb{R}^{nk}$.

Here, we introduce the bijections $can_{tr} : ((\mathbb{R}^2)^3)^k \to (\mathbb{R}^2)^{3k}$, mapping k-tuples of triangles to $(3k)$ tuples of points in $\mathbb{R}^2$ and $can_{trST} : ((\mathbb{R}^2 \times \mathbb{R})^3)^k \to (\mathbb{R}^2 \times \mathbb{R})^{3k}$, mapping k-tuples of spatio-temporal triangles to $(3k)$-tuples of points in $(\mathbb{R}^2 \times \mathbb{R})$.

Definition 7.2 (**Triangle relations and databases**). A *(triangle) database schema* $\hat{\sigma}$ is a finite set of relation names, where each relation name $\hat{R}$ has a natural number $ar(\hat{R})$, called its arity, associated to it.

• A subset $\mathcal{C}$ of $((\mathbb{R}^2)^3)^k$ is a *spatial triangle relation of arity k* if

(i) its image under the canonical bijection $can \circ can_{tr} : ((\mathbb{R}^2)^3)^k \to \mathbb{R}^{6k}$ is a semi-algebraic relation of arity $6k$, and

(ii) for each element $c = ((\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}), (\mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}), \ldots, (\mathbf{a}_{k,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3})) \in \mathcal{C}$, also the elements $((\mathbf{a}_{1,j_{1,1}}, \mathbf{a}_{1,j_{1,2}}, \mathbf{a}_{1,j_{1,3}}), (\mathbf{a}_{2,j_{2,1}}, \mathbf{a}_{2,j_{2,2}}, \mathbf{a}_{2,j_{2,3}}), \ldots, (\mathbf{a}_{k,j_{k,1}}, \mathbf{a}_{k,j_{k,2}}, \mathbf{a}_{k,j_{k,3}}))$ are in $\mathcal{C}$, where $\sigma_i(1,2,3) = (j_{i,1}, j_{i,2}, j_{i,3})$ with $1 \leq i \leq k$ and $\sigma_i \in \mathcal{S}_3$ where $\mathcal{S}_3$ is the set of all permutations of $\{1,2,3\}$.

Let $\hat{\sigma}$ be a triangle database schema. A *spatial triangle database* over $\hat{\sigma}$ in $(\mathbb{R}^2)^3$ is a structure $\mathcal{D}$ over $\hat{\sigma}$ with domain $(\mathbb{R}^2)^3$ such that, for each relation name $\hat{R}$ of $\hat{\sigma}$, the associated triangle relation $\hat{R}^{\mathcal{D}}$ in $\mathcal{D}$ is a spatial triangle relation of arity $ar(\hat{R})$.

- A subset $\mathcal{C}$ of $((\mathbb{R}^2 \times \mathbb{R})^3)^k$ is a *spatio-temporal triangle relation of arity k* if

  (i) its image under the canonical bijection $can_{trST} \circ can_{ST} : ((\mathbb{R}^2 \times \mathbb{R})^3)^k \to \mathbb{R}^{9k}$ is a semi-algebraic relation of arity $9k$, and

  (ii) for each element $c = ((p_{1,1}, p_{1,2}, p_{1,3}), (p_{2,1}, p_{2,2}, p_{2,3}), \ldots, (p_{k,1}, p_{k,2}, p_{k,3})) \in \mathcal{C}$, also $((p_{1,j_{1,1}}, p_{1,j_{1,2}}, p_{1,j_{1,3}}), (p_{2,j_{2,1}}, p_{2,j_{2,2}}, p_{2,j_{2,3}}), \ldots, (p_{k,j_{k,1}}, p_{k,j_{k,2}}, p_{k,j_{k,3}}))$ are in $\mathcal{C}$, where $\sigma_i(1,2,3) = (j_{i,1}, j_{i,2}, j_{i,3})(1 \leq i \leq k; \sigma_i \in \mathcal{S}_3)$. Here, $\mathcal{S}_3$ is the set of all permutations of $\{1,2,3\}$.

Let $\hat{\sigma}^{st}$ be a triangle database schema. A spatio-temporal *triangle database* over $\hat{\sigma}^{st}$ is a structure $\mathcal{D}^{st}$ over $\hat{\sigma}^{st}$ with domain $(\mathbb{R}^2 \times \mathbb{R})^3$ such that, for each relation name $\hat{R}^{st}$ of $\hat{\sigma}^{st}$, the associated triangle relation $\hat{R}^{st\mathcal{D}^{st}}$ in $\mathcal{D}^{st}$ is a spatio-temporal triangle relation of arity $ar(\hat{R}^{st})$.

We want to remark two things about the definition of triangle relations (as given in Definition 7.2), one about the items (i) and one about the items (ii) of the definition of triangle relations. They are discussed in Remark 7.3 below and Remark 7.6, which is postponed until after the definition of triangle database queries.

Remark 7.3. A triangle database $\mathcal{D}$ over $\hat{\sigma}$ in $(\mathbb{R}^2)^3$ can be viewed naturally as a geometric database $\mathcal{S}$ over the schema $\dot{\sigma}$, which has, for each relation name $\hat{R}$ of $\hat{\sigma}$, a relation name $\dot{R}$ with arity $3 \times ar(\hat{R})$. For each relation name $\hat{R}$, of arity k, $\dot{R}^{\mathcal{S}}$ is obtained from $\hat{R}^{\mathcal{D}}$ by applying the canonical bijection $can_{tr} : ((\mathbb{R}^2)^3)^k \to (\mathbb{R}^2)^{3k}$. Analogously, a spatio-temporal triangle database $\mathcal{D}^{st}$ over $\hat{\sigma}^{st}$ can be viewed naturally as a spatio-temporal database $\mathfrak{F}$ over the schema $\dot{\sigma}^{st}$, which has, for each relation name $\hat{R}^{st}$ of $\hat{\sigma}^{st}$, a relation name $\dot{R}^{st}$ with arity $3 \times ar(\hat{R}^{st})$. For each relation name $\hat{R}^{st}$, of arity k, $\dot{R}^{st\mathfrak{F}}$ is obtained from $\hat{R}^{st\mathcal{D}^{st}}$ by applying the canonical bijection $can_{trST} : ((\mathbb{R}^2 \times \mathbb{R})^3)^k \to (\mathbb{R}^2 \times \mathbb{R})^{3k}$.

Example 7.4. It follows from the definition of triangle relations that they can be finitely represented by polynomial constraints on the coordinates of their corner points.

For example, the unary spatial triangle relation containing all triangles with one corner point on the x-axis, one on the y-axis and one on the diagonal $y = x$, can be finitely represented as follows:

$$\begin{aligned}\{(\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3) = ((a_{1,x}, a_{1,y}), (a_{2,x}, a_{2,y}), (a_{3,x}, a_{3,y})) \in (\mathbb{R}^2)^3 \ \mid \\ (a_{1,x} = 0 \wedge a_{2,y} = 0 \wedge a_{3,x} = a_{3,y}) \vee (a_{1,x} = 0 \wedge a_{3,y} = 0 \wedge a_{2,x} = a_{2,y}) \\ \vee (a_{2,x} = 0 \wedge a_{1,y} = 0 \wedge a_{3,x} = a_{3,y}) \vee (a_{2,x} = 0 \wedge a_{3,y} = 0 \wedge a_{1,x} = a_{1,y}) \\ \vee (a_{3,x} = 0 \wedge a_{2,y} = 0 \wedge a_{1,x} = a_{1,y}) \vee (a_{3,x} = 0 \wedge a_{1,y} = 0 \wedge a_{2,x} = a_{2,y})\}.\end{aligned}$$

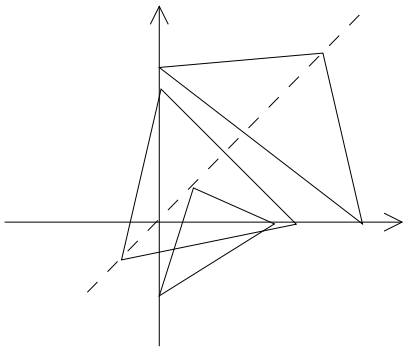


Figure 7.1: Some elements of the relation represented in Example 7.4.

Figure 7.1 gives some elements of this relation. Each triangle that is drawn is stored three times in the relation.

□

We now define spatial and spatio-temporal triangle database queries.

Definition 7.5 (**Triangle database queries**). • Let $\hat{\sigma}$ be a triangle database schema and let us consider input spatial triangle databases over $\hat{\sigma}$. A *k-ary spatial triangle database query* Q over $\hat{\sigma}$ is a computable partial mapping (in the sense of Remark 3.5) from the set of spatial triangle databases over $\hat{\sigma}$ to the set of k-ary spatial triangle relations.

• Let $\hat{\sigma}^{st}$ be a database schema and let us consider input spatio-temporal triangle databases over $\hat{\sigma}^{st}$. A *k-ary* spatio-temporal *triangle database query* Q over $\hat{\sigma}^{st}$ is a computable partial mapping (in the sense of Remark 3.5) from the set of spatio-temporal triangle databases over $\hat{\sigma}^{st}$ to the set of k-ary spatio-temporal triangle relations.

Remark 7.6. In the (ii)-items of the definition of triangle relations, we require that, if a triangle T is involved in a relation, that also all other triangles with the same drawing are stored in that relation. The reason for this is that we do not want the triangle queries to be dependent of the actual order and orientation used when enumerating the corner points of a triangle. When emphasizing property (ii) of a relation, we will call it *consistency* and talk about *consistent triangle relations*. Also, a database is said to be consistent, if all its relations are consistent.

We illustrate the consistency property with some examples:

Example 7.7. Let $\hat{\sigma} = \{\hat{R}\}$ be a database schema. First, we list some queries over $\hat{\sigma}$ that are not consistent:

• *Q_6: Give all triangles in $\hat{R}$ for which their first and second corner points coincide.*

• *Q_7: Give all triangles for which the segment defined by their first and second corner point is a boundary segment of one of the triangles in $\hat{R}$.*

Now some consistent queries follow:

- Q_8: *Give all triangles in $\hat{R}$ that are degenerated into a line segment.*
- Q_9: *Give all triangles that share a boundary segment with some triangle in $\hat{R}$.*

It is clear that the inconsistent queries are rather artificial. When a user specifies the triangles that should be in the result of a query, she intuitively thinks of the drawings of those triangles. The order of the corner points used in the construction of those triangles should not be important. □

In Chapter 3 we remarked that spatial and spatio-temporal database queries could be naturally interpreted as constraint queries. Analogously, spatial and spatio-temporal triangle database queries can be seen as constraint queries, and as spatial and spatio-temporal (point) database queries. We prefer the latter view, as we already developed affine-generic spatio-temporal point languages in Chapter 4, and there already exist affine-generic spatial point languages, as we described in Chapter 3. We define the equivalence between triangle queries and point queries formally:

Definition 7.8 (**Equivalence of point queries and triangle queries**). • Let $\hat{\sigma}$ be a triangle database schema and let us consider input spatial triangle databases over $\hat{\sigma}$. Let $\dot{\sigma}$ be the corresponding spatial point database schema (see Remark 7.3). Let $\hat{Q}$ be a k-ary spatial triangle database query over $\hat{\sigma}$ and let $\dot{Q}$ be a $(3k)$-ary spatial (point) database query over $\dot{\sigma}$. We say that $\hat{Q}$ *and* $\dot{Q}$ *are equivalent*, denoted $\hat{Q} \equiv_{\triangle} \dot{Q}$ if for every database $\mathcal{D}$ over $\hat{\sigma}$ we have

$$can_{tr}(\hat{Q}(\mathcal{D})) = \dot{Q}(can_{tr}(\mathcal{D})).$$

• Let $\hat{\sigma}^{st}$ be a triangle database schema and let us consider input spatio-temporal triangle databases over $\hat{\sigma}^{st}$. Let $\dot{\sigma}^{st}$ be the corresponding spatio-temporal point database schema (see Remark 7.3). Let $\hat{Q}$ be a k-ary spatio-temporal triangle database query over $\hat{\sigma}^{st}$ and let $\dot{Q}$ be a $(3k)$-ary spatio-temporal (point) database query over $\dot{\sigma}^{st}$. We say that $\hat{Q}$ *and* $\dot{Q}$ *are equivalent*, denoted $\hat{Q} \equiv_{\triangle} \dot{Q}$, if for every database $\mathcal{D}^{st}$ over $\hat{\sigma}^{st}$ we have

$$can_{trST}(\hat{Q}(\mathcal{D}^{st})) = \dot{Q}(can_{trST}(\mathcal{D}^{st})).$$

Since we have defined equivalence between triangle database queries and point database queries, we can now discuss how the point languages FO($\{\mathbf{Between}^{(2)}\}$) and FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$) can be used to query triangle databases. We have to keep in mind that only spatial and spatio-temporal (point) databases can be considered that are the image under the bijections can_{tr} and can_{trST} of spatial and spatio-temporal triangle databases.

Definition 7.9 (FO($\{$**Between**$\}$) **as a triangle query language**). • Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\dot{R}_i$ be the corresponding spatial point relation names of arity $3 \times ar(\hat{R}_i)$, for $i = 1 \ldots m$, and let $\dot{\sigma}$ be the spatial database schema $\{\dot{R}_1, \dot{R}_2, \ldots, \dot{R}_m\}$.

Let $\varphi(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}, \mathbf{x}_{2,1}, \mathbf{x}_{2,2}, \mathbf{x}_{2,3}, \ldots, \mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \mathbf{x}_{k,3})$ be a FO($\{$**Between**$\}$) - formula, expressing a spatial $(3k)$-ary query $\dot{Q}$ which is equivalent to a k-ary spatial triangle query $\hat{Q}$. For each input spatial triangle database $\mathcal{D}$ over $\hat{\sigma}$, $\hat{Q}(\mathcal{D})$ is defined as the set of points $(\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}, \mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}, \ldots, \mathbf{a}_{k,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3})$ in $(\mathbb{R}^6)^k$

such that

$$(\mathbb{R}^2, =, \mathbf{Between}^{(2)}, \dot{R}_1^{\mathcal{S}}, \dot{R}_2^{\mathcal{S}}, \ldots, \dot{R}_m^{\mathcal{S}}) \models \varphi[\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}, \mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}, \ldots, \mathbf{a}_{k,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3}].$$

Here, $\mathcal{S}$ is the image of $\mathcal{D}$ under the canonical bijection can_{tr}.

• Let $\hat{\sigma}^{st} = \{\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}\}$ be a spatio-temporal triangle database schema. Let $\dot{R}_i^{st} (1 \leq i \leq m)$ be the corresponding spatio-temporal point relation names of arity $3 \times ar(\hat{R}_i^{st})$ and let $\dot{\sigma}^{st}$ be the spatio-temporal database schema $\{\dot{R}_1^{st}, \dot{R}_2^{st}, \ldots, \dot{R}_m^{st}\}$.

Let $\varphi(u_{1,1}, u_{1,2}, u_{1,3}, u_{2,1}, u_{2,2}, u_{2,3}, \ldots, u_{k,1}, u_{k,2}, u_{k,3})$ be a FO($\{$**Between**$\}$) -formula, expressing a spatio-temporal $(3k)$-ary query $\dot{Q}$ which is equivalent to a k-ary spatial triangle query $\hat{Q}$. For each input spatio-temporal triangle database $\mathcal{D}^{st}$ over $\hat{\sigma}^{st}$, $\hat{Q}(\mathcal{D}^{st})$ is defined as the set of points $(p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{2,3}, \ldots, p_{k,1}, p_{k,2}, p_{k,3})$ of $(\mathbb{R}^9)^k$ such that

$$(((\mathbb{R}^2 \times \mathbb{R})), =, \mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}, \dot{R}_1^{st\mathcal{D}^{st}}, \dot{R}_2^{st\mathcal{D}^{st}}, \ldots, \dot{R}_m^{st\mathcal{D}^{st}}) \models \varphi[p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{2,3}, \ldots, p_{k,1}, p_{k,2}, p_{k,3}].$$

Here, $\mathfrak{F}$ is the image of $\mathcal{D}^{st}$ under the canonical bijection can_{trST}.

The languages FO($\{$**Between**$^{(2)}\}$, $\dot{\sigma}$) and FO($\{$**Between**, **Before**, **EqCr**$^{ST}\}$, $\dot{\sigma}^{st}$) were designed to formulate queries on spatial and spatio-temporal point databases over some input schema $\dot{\sigma}$, resp. $\dot{\sigma}^{st}$. Using those languages to query triangle databases, involves expressing relations between the point sets that compose the triangles. This is a rather indirect way of expressing triangle relations. In the spirit of Chapter 4, we now construct affine-generic query languages based on triangle variables. As they directly express relations between the triangles, this results in a more intuitive way of querying spatial and spatio-temporal triangle databases. We define triangle-based logics next. Afterwards, we propose a specific spatial triangle logic in Section 7.3, and a spatio-temporal triangle logic in Section 7.4.

Definition 7.10 (**Triangle logics**). • Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a triangle database schema and let $\mathbf{\Delta}$ be a set of predicates of a certain arity over triangles in $\mathbb{R}^2$. The first-order logic over $\hat{\sigma}$ and $\mathbf{\Delta}$, denoted by FO($\mathbf{\Delta}$, $\hat{\sigma}$), can be used as a spatial triangle query language when variables are interpreted to range over triangles in $\mathbb{R}^2$. The atomic formulas in FO($\mathbf{\Delta}$, $\hat{\sigma}$) are equality constraints on triangle variables, the predicates of $\mathbf{\Delta}$, and the relation names $\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m$ from $\hat{\sigma}$, applied to triangle variables.

• Let $\hat{\sigma}^{st} = \{\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}\}$ be a database schema and let $\mathbf{\Delta}$ be a set of predicates of a certain arity over spatio-temporal triangles in $(\mathbb{R}^2 \times \mathbb{R})$. The first-order logic over $\hat{\sigma}^{st}$ and $\mathbf{\Delta}$, denoted by FO($\mathbf{\Delta}$, $\hat{\sigma}^{st}$), can be used as a spatio-temporal triangle query language when variables are interpreted to range over spatio-temporal triangles in $(\mathbb{R}^2 \times \mathbb{R})$. The atomic formulas in FO($\mathbf{\Delta}$, $\hat{\sigma}^{st}$) are equality constraints on spatio-temporal triangle variables, the predicates of $\mathbf{\Delta}$, and the relation names $\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}$ from $\hat{\sigma}^{st}$, applied to spatio-temporal triangle variables.

A FO($\mathbf{\Delta}$, $\hat{\sigma}$)-formula $\varphi(\triangle_1, \triangle_2, \ldots, \triangle_k)$ (resp., FO($\mathbf{\Delta}$, $\hat{\sigma}^{st}$)-formula $\varphi(\triangle_1^{st}, \triangle_2^{st}, \ldots, \triangle_k^{st})$) defines for each spatial (resp., spatio-temporal) database $\mathcal{D}$ (resp., $\mathcal{D}^{st}$)

over $\hat{\sigma}$ (resp. $\hat{\sigma}^{st}$) a subset $\varphi(\mathcal{D})$ (resp., $\varphi(\mathcal{D}^{st})$) of $\left(\left(\mathbb{R}^2\right)^3\right)^k$ (resp., $\left(\left(\mathbb{R}^2 \times \mathbb{R}\right)^3\right)^k$) defined as

$$\{(T_1, T_2, \ldots, T_k) \in \left(\mathbb{R}^2\right)^{3k} \mid (\mathbb{R}^2, \mathbf{\Delta}^{\mathbb{R}^2}, \hat{R}_1^{\mathcal{D}}, \hat{R}_2^{\mathcal{D}}, \ldots, \hat{R}_m^{\mathcal{D}}) \models \varphi[T_1, T_2, \ldots, T_k] \},$$

respectively,

$$\{(T_1^{st}, T_2^{st}, \ldots, T_k^{st}) \in \left(\mathbb{R}^2 \times \mathbb{R}\right)^{3k} \mid ((\mathbb{R}^2 \times \mathbb{R}), \mathbf{\Delta}^{(\mathbb{R}^2 \times \mathbb{R})}, \hat{R}_1^{st\mathcal{D}^{st}}, \hat{R}_2^{st\mathcal{D}^{st}}, \ldots, \hat{R}_m^{st\mathcal{D}^{st}}) \models \varphi[T_1^{st}, T_2^{st}, \ldots, T_k^{st}] \}.$$

Remark 7.11. We use the symbol $=_{\Delta}$ to indicate equality of triangle variables, as opposed to equality of point variables. If it is clear from the context of a formula which type of variables is used, we will omit the index.

In Section 7.3 (resp., Section 7.4), we will develop languages that have the same expressive power as FO($\{$**Between**$\}$) and FO($\{$**Between**$^{\text{Cotemp}}$, **Before**, **EqCr**$^{ST}\}$) on spatial triangle databases and on spatio-temporal triangle databases, respectively. We will prove this by showing both soundness and completeness of those triangle languages with respect to FO($\{$**Between**$\}$) and FO($\{$**Between**$^{\text{Cotemp}}$, **Before**, **EqCr**$^{ST}\}$). The concepts of soundness and completeness were introduced in Definition 4.12

7.3 Affine-invariant Spatial Triangle Queries

In this section, we propose a spatial triangle logic that captures exactly the class of first-order affine-generic queries on spatial triangle databases. First, we remark the following:

Remark 7.12. We defined a triangle database as a special type of geometric database. Accordingly, we take the affine image of a triangle for affinities of $\mathbb{R}^2$, and not of $\mathbb{R}^6$. This corresponds to our intuition. One triangle is an affine image of another triangle, if the drawing of the first one is the affine image of the drawing of the second one. Hence, the affine image of a triangle with corner points $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$ under some affinity α of the plane, is the triangle with corner points $\alpha(\mathbf{x}_1)$, $\alpha(\mathbf{x}_2)$ and $\alpha(\mathbf{x}_3)$.

We introduce one binary triangle predicate, *i.e.*, **PartOf**. Intuitively, when applied to two triangles, this predicate expresses that the drawing of the first triangle is a subset ($\subseteq$) of the drawing of the second triangle. We only consider $\left(\mathbb{R}^2\right)^3$ as the underlying domain. We show that the triangle predicate **PartOf** allows a natural extension to higher dimensions and other types of objects (instead of triangles).

We define the predicate **PartOf** and equality on triangles more precisely:

Definition 7.13 (**The triangle predicate PartOf**)**.** Let $T_1 = (\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3})$ and $T_2 = (\mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3})$ be two triangles. The binary predicate **PartOf**, applied to T_1 and T_2 expresses that the convex closure of the three points $\mathbf{a}_{1,1}$, $\mathbf{a}_{1,2}$ and $\mathbf{a}_{1,3}$ is a subset of the convex closure of the three points $\mathbf{a}_{2,1}$, $\mathbf{a}_{2,2}$ and $\mathbf{a}_{2,3}$.

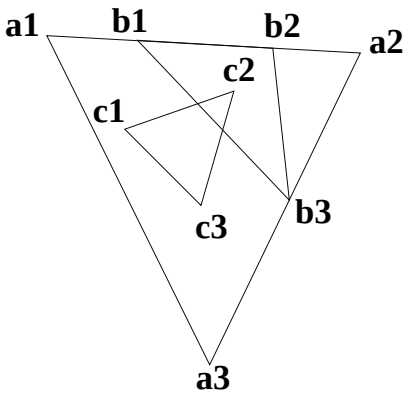


Figure 7.2: An illustration of the predicate **PartOf**. Let $T_1 = (\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$, $T_2 = (\mathbf{b}_1, \mathbf{b}_2, \mathbf{b}_3)$ and $T_3 = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ The expressions $\mathbf{PartOf}(T_2, T_1)$ and $\mathbf{PartOf}(T_3, T_1)$ are true, the expression $\mathbf{PartOf}(T_3, T_2)$ is not true.

Figure 7.2 illustrates the predicate **PartOf**.

We also define triangle-equality, which differs from the standard meaning of equality.

Definition 7.14 (**Equality of triangles**)**.** Let T_1 and T_2 be two triangles. The expression $T_1 =_\triangle T_2$ is true iff both $\mathbf{PartOf}(T_1, T_2)$ and $\mathbf{PartOf}(T_2, T_1)$ are tue.

From now on, in the remainder of this section, we will assume that

$$\mathbf{\Delta} = \{\mathbf{PartOf}\}.$$

Before analyzing the expressiveness of the language FO($\mathbf{\Delta}$), we prove that the FO($\mathbf{\Delta}$)-queries are well-defined on consistent triangle databases. More concretely, given a triangle database schema $\hat{\sigma}$, we prove that the result of a k-ary FO(Δ, $\hat{\sigma}$) query on a consistent input database over $\hat{\sigma}$ is a consistent triangle relation of arity k.

Lemma 7.15 (**FO($\mathbf{\Delta}$) is well-defined**)**.** Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\mathcal{D}$ be a consistent spatial triangle database over $\hat{\sigma}$. For each FO(Δ, $\hat{\sigma}$)-query $\hat{Q}$, $\hat{Q}(\mathcal{D})$ is a consistent triangle relation.

Proof. Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\mathcal{D}$ be a consistent spatial triangle database over $\hat{\sigma}$.

We prove this lemma by induction on the structure of FO(Δ, $\hat{\sigma}$)-queries. The atomic formulas of FO($\mathbf{\Delta}$) are equality expressions on triangle variables, expressions of the form $\mathbf{PartOf}(\triangle_1, \triangle_2)$, and expressions of the form $\hat{R}_i(\triangle_1, \triangle_2, \ldots, \triangle_{ar(\hat{R}_i)})$, where $\hat{R}_i (1 \leq i \leq m)$ is a relation name from $\hat{\sigma}$. More complex formulas can be constructed using the Boolean operators $\wedge$, $\vee$ and $\neg$ and existential quantification.

For the atomic formulas, it is easy to see that, if two triangles T_1 and T_2 satisfy the conditions $T_1 =_\triangle T_2$ or $\mathbf{PartOf}(T_1, T_2)$, that also $T_1' =_\triangle T_2'$ respectively

PartOf(T_1', T_2') are true iff $T_1 =_{\triangle} T_1'$ and $T_2 =_{\triangle} T_2'$ are true. As we assume the input database $\mathcal{D}$ to be consistent, the atomic formulas of the type $\hat{R}_i(\triangle_1, \triangle_2, \ldots, \triangle_{ar(\hat{R}_i)})$, where $(1 \leq i \leq m)$, trivially return consistent triangle relations.

Now we have to prove that the composed formulas always return consistent triangle relations. Let $\hat{\varphi}$ and $\hat{\psi}$ be two formulas in FO(Δ, $\hat{\sigma}$), of arity k_φ and k_ψ respectively, already defining consistent triangle relations. Then, the formula $(\hat{\varphi} \wedge \hat{\psi})$ (resp., $(\hat{\varphi} \vee \hat{\psi})$) also defines a triangle relation. This follows from the fact that the free variables of $(\hat{\varphi} \wedge \hat{\psi})$ (resp., $(\hat{\varphi} \vee \hat{\psi})$) are free variables in $\hat{\varphi}$ or $\hat{\psi}$. The universe of all triangles is trivially consistent. If a consistent subset is removed from this universe, the remaining part is still consistent. Therefor, $\neg\hat{\varphi}$ is well-defined. Finally, because consistency is defined argument-wise, the projection $\exists T_1\, \hat{\varphi}(T_1, T_2, \ldots, T_{k_\varphi})$ is consistent. □

After proving that the language FO($\boldsymbol{\Delta}$) is well-defined, we can analyze its expressiveness.

7.3.1 Expressiveness of FO($\boldsymbol{\Delta}$)

We now determine the expressiveness of the language FO($\boldsymbol{\Delta}$). We prove that it is sound and complete for the affine-invariant fragment of first-order logic over the reals, on triangle databases. We prove this by comparing the languages FO($\boldsymbol{\Delta}$) and FO($\{\mathbf{Between}^{(2)}\}$). From Chapter 3, we already know that FO($\{\mathbf{Between}\}$) is sound and complete for the affine-invariant fragment of first-order logic over the reals, on spatial point databases.

The soundness and completeness of the query language FO($\boldsymbol{\Delta}$) with respect to the language FO($\{\mathbf{Between}^{(2)}\}$) is proved using two separate lemmas (Lemma 7.16 and Lemma 7.17). In both lemmas, formulas are translated from one language in the other, by using induction on the structure of FO($\boldsymbol{\Delta}$) and FO($\{\mathbf{Between}^{(2)}\}$)-formulas, respectively. This proof technique will be used several times in this chapter. Therefor, we explain the first such proofs in detail. Later on, we will only develop the crucial points in similar proofs.

Lemma 7.16 (soundness of FO($\boldsymbol{\Delta}$)**with respect to** FO($\{\mathbf{Between}^{(2)}\}$)**).** Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\dot{R}_i$ be the corresponding spatial point relation names of arity $3 \times ar(\hat{R}_i)$, for $(1 \leq i \leq m)$, and let $\dot{\sigma}$ be the spatial database schema $\{\dot{R}_1, \dot{R}_2, \ldots, \dot{R}_m\}$. Every FO($\Delta$, $\hat{\sigma}$)-expressible query can be expressed equivalently in FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$).

Proof. Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\dot{R}_i$ be the corresponding spatial point relation names of arity $3 \times ar(\hat{R}_i)$, for $(1 \leq i \leq m)$, and let $\dot{\sigma}$ be the corresponding spatial database schema $\{\dot{R}_1, \dot{R}_2, \ldots, \dot{R}_m\}$. We translate each formula of FO(Δ, $\hat{\sigma}$) into an equivalent formula in FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$). We do this by induction on the structure of FO(Δ, $\hat{\sigma}$)-formulas.

First, we translate the variables of $\hat{\varphi}$. Each triangle variable $\triangle$ is naturally translated into three spatial point variables $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. We allow one or more of the corner points of a triangle to coincide, so there are no further restrictions on the variables $\mathbf{x}_j, 1 \leq j \leq 3$.

The atomic formulas of FO(Δ, $\hat{\sigma}$) are equality expressions on triangle variables, expressions of the form **PartOf**($\triangle_1, \triangle_2$), and expressions of the form $\hat{R}_i(\triangle_1, \triangle_2, \ldots, \triangle_k)$, where $k = ar(\hat{R}_i)$ and $1 \leq i \leq m$. More complex formulas can be constructed using the Boolean operators $\wedge$, $\vee$ and $\neg$ and existential quantification.

The translation of atomic formulas.

We first show that all atomic formulas of FO(Δ, $\hat{\sigma}$) can be expressed in the language FO($\{$**Between**$\}$, $\dot{\sigma}$) .

(i) The translation of ($\triangle_1$=$\triangle_2$), where T_1 is translated into $\mathbf{x}_{1,1}$, $\mathbf{x}_{1,2}$ and $\mathbf{x}_{1,3}$ and T_2 is translated into $\mathbf{x}_{2,1}$, $\mathbf{x}_{2,2}$ and $\mathbf{x}_{2,3}$, equalsLet $T_1^{st} = (p_{1,1}, p_{1,2}, p_{1,3})$ and $T_2^{st} = (p_{2,1}, p_{2,2}, p_{2,3})$ be two triangle snapshots. The binary predicate **PartOf**, applied to T_1^{st} and T_2^{st} expresses that $p_{1,1}$, $p_{1,2}$ and $p_{1,3}$ (resp., $p_{2,1}$, $p_{2,2}$ and $p_{2,3}$) are co-temporal and that the convex closure of the three points $p_{1,1}$, $p_{1,2}$ and $p_{1,3}$ is a subset of the convex closure of the three points $p_{2,1}$, $p_{2,2}$ and $p_{2,3}$. □

$$\bigvee_{\sigma(1,2,3)=(j_1,j_2,j_3),\sigma\in\mathcal{S}_3} (\mathbf{x}_{1,1} = \mathbf{x}_{2,j_1} \wedge \mathbf{x}_{1,2} = \mathbf{x}_{2,j_2} \wedge \mathbf{x}_{1,3} = \mathbf{x}_{2,j_3}),$$

where $\mathcal{S}_3$ is the set of all permutations of $\{1, 2, 3\}$.

The correctness of this translation follows trivially from the definition of triangle equality (see Definition 7.14).

(ii) The translation of **PartOf**($\triangle_1, \triangle_2$), where T_1 is translated into $\mathbf{x}_{1,1}$, $\mathbf{x}_{1,2}$ and $\mathbf{x}_{1,3}$ and T_2 is translated into $\mathbf{x}_{2,1}$, $\mathbf{x}_{2,2}$ and $\mathbf{x}_{2,3}$, is

$$\bigwedge_{i=1}^{3} \mathbf{InTriangle}(\mathbf{x}_{1,i}, \mathbf{x}_{2,1}, \mathbf{x}_{2,2}, \mathbf{x}_{2,3}),$$

where the definition of **InTriangle** is:

$$\mathbf{InTriangle}(\mathbf{x}, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3) := \exists\, \mathbf{x}_4(\mathbf{Between}^{(2)}(\mathbf{x}_1, \mathbf{x}_4, \mathbf{x}_2) \wedge \mathbf{Between}^{(2)}(\mathbf{x}_4, \mathbf{x}, \mathbf{x}_3)).$$

Figure 7.3 illustrates the corresponding geometric construction.

The correctness of this translation follows from the definition of the predicate **PartOf** (see Definition 7.13).

(iii) The translation of $\hat{R}_i(\triangle_1, \triangle_2, \ldots, \triangle_k)$, where T_i is translated into $\mathbf{x}_{i,1}$, $\mathbf{x}_{i,2}$ and $\mathbf{x}_{i,3}$ for $1 \leq i \leq k$, is

$$\dot{R}_i(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}, \mathbf{x}_{2,1}, \mathbf{x}_{2,2}, \mathbf{x}_{2,3}, \ldots, \mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \mathbf{x}_{k,3}).$$

The correctness of this translation follows from Definition 7.2 and Remark 7.3.

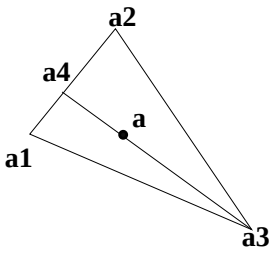


Figure 7.3: An illustration of the predicate **InTriangle**. The expression $\mathbf{InTriangle}(\mathbf{a}, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3)$ is true because there exists a point $\mathbf{a}_4$ between $\mathbf{a}_1$ and $\mathbf{a}_2$ such that $\mathbf{a}$ lies between $\mathbf{a}_4$ and $\mathbf{a}_3$.

The translation of composed formulas.

Assume that we already correctly translated the FO(Δ, $\hat{\sigma}$)-formulas $\hat{\varphi}$ and $\hat{\psi}$ into the FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-formulas $\dot{\varphi}$ and $\dot{\psi}$. Suppose that the number of free variables in $\hat{\varphi}$ is k_φ and that of $\hat{\psi}$ is k_ψ. Therefor, we can assume that, for each triangle database $\mathcal{D}$ over the input schema $\hat{\sigma}$, and for each k_φ-tuple of triangles $(T_1, T_2, \ldots, T_{k_\varphi})$ given as $((\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}), (\mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}), \ldots, (\mathbf{a}_{k_\varphi,1}, \mathbf{a}_{k_\varphi,2}, \mathbf{a}_{k_\varphi,3}))$ that

$$\mathcal{D} \models \hat{\varphi}(T_1, T_2, \ldots, T_{k_\varphi}) \text{ if and only if } \mathcal{S} \models \dot{\varphi}(\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}, \mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}, \ldots, \mathbf{a}_{k_\varphi,1}, \mathbf{a}_{k_\varphi,2}, \mathbf{a}_{k_\varphi,3})$$

is true when $\mathcal{S}$ is the spatial (point) database over the input schema $\dot{\sigma}$, obtained from $\mathcal{D}$ by applying the canonical bijection can_{tr} between $\left((\mathbb{R}^2)^3\right)^{k_\varphi}$ and $(\mathbb{R}^2)^{3k_\varphi}$, on $\mathcal{D}$. For the formula $\hat{\psi}$ the analog holds.

In the following, we omit the k_φ-tuples (resp., k_ψ-tuples) of triangles and $3k_\varphi$-tuples (resp., $3k_\psi$-tuples) of points the formulas are applied on, to make the proofs more readable.

(i) The translation of $\hat{\varphi} \wedge \hat{\psi}$ is $\dot{\varphi} \wedge \dot{\psi}$. Indeed,

$\mathcal{S} \models (\dot{\varphi} \wedge \dot{\psi})$
iff. $\mathcal{S} \models \dot{\varphi}$ and $\mathcal{S} \models \dot{\psi}$
iff. $\mathcal{D} \models \hat{\varphi}$ and $\mathcal{D} \models \hat{\psi}$
iff. $\mathcal{D} \models (\hat{\varphi} \wedge \hat{\psi})$.

(ii) The translation of $\hat{\varphi} \vee \hat{\psi}$ is $\dot{\varphi} \vee \dot{\psi}$. Indeed,

$\mathcal{S} \models (\dot{\varphi} \vee \dot{\psi})$
iff. $\mathcal{S} \models \dot{\varphi}$ or $\mathcal{S} \models \dot{\psi}$
iff. $\mathcal{D} \models \hat{\varphi}$ or $\mathcal{D} \models \hat{\psi}$
iff. $\mathcal{D} \models (\hat{\varphi} \vee \hat{\psi})$.

(iii) The translation of $\neg\hat{\varphi}$ is $\neg\dot{\varphi}$. Indeed,

$\mathcal{S} \models \neg\dot{\varphi}$
iff. it is not true that $\mathcal{S} \models \dot{\varphi}$
iff. it is not true that $\mathcal{D} \models \hat{\varphi}$
iff. $\mathcal{D} \models \neg\hat{\varphi}$.

(iv) Assume that $\hat{\varphi}$ has free variables $\triangle, \triangle_1, \ldots, \triangle_k$ and $\triangle$ is translated into $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and $\triangle_i$ is translated into $\mathbf{x}_{i,1}$, $\mathbf{x}_{i,2}$, $\mathbf{x}_{i,3}$. The translation of

$$\exists\, \triangle\ \ \hat{\varphi}(\triangle, \triangle_1, \triangle_2, \ldots, \triangle_k)$$

$$\text{is } \exists\mathbf{x}_1\, \exists\mathbf{x}_2\, \exists\mathbf{x}_3\, \dot{\varphi}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}, \ldots, \mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \mathbf{x}_{k,3}).$$

Indeed,

$\mathcal{S} \models$
$\exists\mathbf{x}_1\, \exists\mathbf{x}_2\, \exists\mathbf{x}_3\, \dot{\varphi}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)[\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}, \ldots, \mathbf{a}_{k,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3}]$
iff. there exist points $\mathbf{a}_1$, $\mathbf{a}_2$, $\mathbf{a}_3$) in $\mathbb{R}^2$ such that
$\mathcal{S} \models \dot{\varphi}[\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}, \ldots, \mathbf{a}_{k,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3}]$
iff. there exists a triangle T such that $\mathcal{D} \models \hat{\varphi}[T, T_1, \ldots, T_k]$, where
T_i is the triangle with corner points $\mathbf{a}_{i,1}$, $\mathbf{a}_{i,2}$ and $\mathbf{a}_{i,3}$ for $1 \leq i \leq k$
iff. $\mathcal{D} \models \exists T\, \hat{\varphi}(T)[T_1, \ldots, T_k]$.

To summarize, let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\dot{\sigma} = \{\dot{R}_1, \dot{R}_2, \ldots, \dot{R}_m\}$ be the corresponding spatial point database schema. Each formula $\hat{\varphi}$ in FO(Δ, $\hat{\sigma}$), with free variables $\triangle_1, \triangle_2, \ldots, \triangle_k$ can be translated into a FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-formula $\dot{\varphi}$ with free variables $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}$, $\mathbf{x}_{2,1}, \mathbf{x}_{2,2}, \mathbf{x}_{2,3}, \ldots, \mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \mathbf{x}_{k,3}$. This translation is such that, for all triangle databases $\mathcal{D}$ over $\hat{\sigma}$, $\mathcal{D} \models \hat{\varphi}$ iff. $\mathcal{S} \models \dot{\varphi}$. Here, $\mathcal{S}$ is the spatial point database over $\dot{\sigma}$ which is the image of $\mathcal{D}$ under the canonical bijection between $\left(\left(\mathbb{R}^2\right)^3\right)^k$ and $\left(\mathbb{R}^2\right)^{3k}$. This completes the soundness proof. □

For completeness, we translate FO($\{\mathbf{Between}^{(2)}\}$)-formulas into FO($\mathbf{\Delta}$)-formulas. We again prove this by induction, this time on the structure of FO($\{\mathbf{Between}^{(2)}\}$)-formulas. This translation is not as straightforward as the translation in the other direction, however.

Lemma 7.17 (Completeness of FO($\mathbf{\Delta}$) with respect to FO($\{\mathbf{Between}\}$)). Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema and $\dot{\sigma}$ be the corresponding spatial database schema. Every FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-expressible query can be expressed equivalently in FO(Δ, $\hat{\sigma}$).

Proof. Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema and $\dot{\sigma}$ be the corresponding spatial database schema. We have to prove that we can translate every triangle database query, expressed in the language FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$), into a triangle database query in the language FO(Δ, $\hat{\sigma}$) over trangle databases.

We first show how we can simulate point variables by a degenerated triangle, and any FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-formula $\dot{\varphi}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ by a formula $\varphi(\triangle_1, \triangle_2, \ldots, \triangle_k)$, where $\triangle_1, \triangle_2, \ldots, \triangle_k$ represent triangles that are degenerated into points. We prove this by induction on the structure of FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-formulas. Initially, each

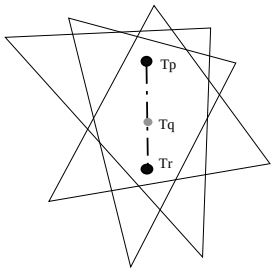


Figure 7.4: Illustration of the translation of the predicate $\mathbf{Between}^{(2)}$. The (degenerated) triangle T_q lies between the (degenerated) triangles T_p and T_r iff all triangles that contain both T_p and T_r, also contain T_q.

FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-formula $\dot{\varphi}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ will be translated into a FO(Δ, $\hat{\sigma}$)-formula $\hat{\varphi}(\triangle_1, \triangle_2, \ldots, \triangle_k)$ with the same number of free variables.

The translation of a point variable $\mathbf{x}$ is the triangle variable $\triangle$, and we add the condition $\mathbf{Point}(\triangle)$ as a conjunct to the beginning of the translation of the formula. The definition of $\mathbf{Point}(\triangle)$ is

$$\forall\, \triangle'\ (\mathbf{PartOf}(\triangle', \triangle) \rightarrow (\triangle{=}\triangle')).$$

In the following, we always assume that such formulas $\mathbf{Point}(\triangle)$ are already added to the translation as a conjunct.

The translation of atomic formulas.

The atomic formulas of the language FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$) are equality constraints on point variables, formulas of the form $\mathbf{Between}^{(2)}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, and formulas of the type $\dot{R}_i(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$, where $k = 3 \times ar(\hat{R}_i)$. We show that all of those can be simulated into an equivalent FO(Δ, $\hat{\sigma}$) formula.

(i) The translation of $(\mathbf{x}_1 = \mathbf{x}_2)$ is $(\triangle_1{=}_{\triangle}\triangle_2)$.

(ii) The translation of $\mathbf{Between}^{(2)}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, where $\triangle_1$, $\triangle_2$ and $\triangle_3$ (which as assumed are already declared points) are the translations of $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$, respectively, is expressed by saying that all triangles that contain both $\triangle_1$ and $\triangle_3$ should also contain $\triangle_2$. It then follows from the convexity of triangles (or line segments, in the degenerated case) that $\triangle_2$ lies on the line segment between $\triangle_1$ and $\triangle_3$. Figure 7.4 illustrates this principle. We now give the formula translating $\mathbf{Between}^{(2)}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$:

$$\forall\, \triangle_4\ ((\mathbf{PartOf}(\triangle_1, \triangle_4) \wedge \mathbf{PartOf}(\triangle_3, \triangle_4)) \rightarrow \mathbf{PartOf}(\triangle_2, \triangle_4)).$$

The correctness of this translation follows from the fact that triangles are convex objects.

(iii) Let $\dot{R}_j$ be a relation name from $\dot{\sigma} = \{\dot{R}_1, \dot{R}_2, \ldots, \dot{R}_m\}$. Let $ar(\hat{R}_j) = k$ and thus $ar(\dot{R}_j) = 3k$, for $1 \leq j \leq m$. The translation of $\dot{R}_j(\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}, \mathbf{x}_{2,1}, \mathbf{x}_{2,2}, \mathbf{x}_{2,3}, \ldots, \mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \mathbf{x}_{k,3})$ is:

$$\exists\, \triangle_1\, \exists\, \triangle_2 \ldots \exists\, \triangle_k\, (\hat{R}_j(\triangle_1, \triangle_2, \ldots, \triangle_k) \wedge \bigwedge_{i=1}^{k} \mathbf{CornerP}(\triangle_{i,1}, \triangle_{i,2}, \triangle_{i,3}, \triangle_i)).$$

The definition of **CornerP** is:

$$\mathbf{CornerP}(\triangle_1, \triangle_2, \triangle_3, \triangle) := \forall\, \triangle_4\ ((\mathbf{Point}(\triangle_4) \wedge \mathbf{PartOf}(\triangle_4, \triangle)) \rightarrow \mathbf{Intriangle}_{\triangle}(\triangle_4, \triangle_1, \triangle_2, \triangle_3)).$$

The predicate $\mathbf{InTriangle}_{\triangle}$ is the translation of the FO($\{\mathbf{Between}^{(2)}\}$) predicate **InTriangle** as described in the proof of Lemma 7.16, into FO($\mathbf{\Delta}$). The FO($\{\mathbf{Between}^{(2)}\}$) formula expressing **InTriangle** only uses $\mathbf{Between}^{(2)}$. In the previous item of this proof, we already showed how this can be translated into FO($\mathbf{\Delta}$).

Given a $(3k)$-tuple of points $(\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}, \mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}, \ldots, \mathbf{a}_{a,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3})$ in $\mathbb{R}^2$. There will be (6^k) k-tuples of triangles $(T_1, T_2, \ldots, T_k)$ such that, for each of the T_i, $1 \leq i \leq k$, the condition $\mathbf{CornerP}(T_{i,1}, T_{i,2}, T_{i,3}, T_i)$ is true. There will, however, only be one tuple of triangles that is the image of the $(3k)$-tuple of points $(\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}, \mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}, \ldots, \mathbf{a}_{a,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3})$ under the inverse of the canonical bijection can_{tr}. Therefor, the simulation is correct.

The translation of composed formulas.

Now suppose that we already simulated the FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$) formulas $\dot{\varphi}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k_\varphi})$ and $\dot{\psi}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{k_\psi})$ into formulas $\hat{\varphi}$ and $\hat{\psi}$ in FO(Δ, $\hat{\sigma}$) with free variables $\triangle_1$, $\triangle_2$, …, $\triangle_{k_\varphi}$ and $\triangle'_1$, $\triangle'_2$, …, $\triangle'_{k_\psi}$, respectively. We can hence assume that, for each triangle database $\mathcal{D}$ over $\hat{\sigma}$ and for each k_φ-tuple of triangles (T_1, T_2, …, T_{k_φ}) = $((\mathbf{a}_1, \mathbf{a}_1, \mathbf{a}_1), (\mathbf{a}_2, \mathbf{a}_2, \mathbf{a}_2), \ldots, (\mathbf{a}_{k_\varphi}, \mathbf{a}_{k_\varphi}, \mathbf{a}_{k_\varphi}))$, which are required to be degenerated into points, that

$$\mathcal{D} \models \hat{\varphi}[T_1, T_2, \ldots, T_{k_\varphi}] \text{ iff. } \mathcal{S} \models \dot{\varphi}[\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{k_\psi}].$$

For $\hat{\psi}$ we have analogue conditions.

The composed formulas $\dot{\varphi} \wedge \dot{\psi}$, $\dot{\varphi} \vee \dot{\psi}$, $\neg\dot{\varphi}$ and $\exists \mathbf{x}\, \dot{\varphi}$, are translated into $\hat{\varphi} \wedge \hat{\psi}$, $\hat{\varphi} \vee \hat{\psi}$, $\neg\hat{\varphi}$ and $\exists\, \triangle\ (\hat{\varphi})$, respectively if we assume that $\mathbf{x}$ is translated into $\triangle$. The correctness proofs for these translations are similar to the proofs in Lemma 7.16. Therefor, we do not repeat them here. This concludes the proof of Lemma 7.17. □

Remark 7.18. So far, we showed that we can simulate any FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$) formula $\dot{\varphi}(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k)$ by a formula $\varphi'(\triangle_1, \triangle_2, \ldots, \triangle_k)$, where $\mathbf{Point}(\triangle_i)$ is true for all $\triangle_i$ $(1 \leq i \leq k)$. If $\dot{\varphi}$ expresses a k-ary triangle database query Q however (*i.e.*, $\dot{\varphi}$ has $(3k)$ free variables), we can do better.

Let $\dot{\varphi}$ be the FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-formula expressing a k-ary triangle database query $\hat{Q}$. The free variables of $\dot{\varphi}$ are $\mathbf{x}_{1,1}, \mathbf{x}_{1,2}, \mathbf{x}_{1,3}, \mathbf{x}_{2,1}, \mathbf{x}_{2,2}, \mathbf{x}_{2,3}, \ldots, \mathbf{x}_{k,1}, \mathbf{x}_{k,2}, \mathbf{x}_{k,3}$.

We now construct the FO(Δ, $\hat{\sigma}$) formula $\hat{\varphi}$ expressing the query $\hat{Q}$ as follows:

$$\begin{aligned}
&\hat{\varphi}(\triangle_1, \triangle_2, \ldots, \triangle_k) \equiv \\
&\quad \exists\, \triangle_{1,1} \exists\, \triangle_{1,2} \exists\, \triangle_{1,3} \exists\, \triangle_{2,1} \exists\, \triangle_{2,2} \exists\, \triangle_{2,3} \ldots \exists\, \triangle_{k,1} \exists\, \triangle_{k,2} \exists\, \triangle_{k,3}\ (\\
&\qquad\qquad \bigwedge_{i=1}^{k} \mathbf{CornerP}(\triangle_{i,1}, \triangle_{i,2}, \triangle_{i,3}, \triangle_i) \wedge \\
&\qquad\qquad \hat{\varphi}'(\triangle_{1,1}, \triangle_{1,2}, \triangle_{1,3}, \triangle_{2,1}, \triangle_{2,2}, \triangle_{2,3}, \ldots, \triangle_{k,1}, \triangle_{k,2}, \triangle_{k,3})),
\end{aligned}$$

For each triple of points, there are 6 different representations for the triangle having those points as its corner points. Therefor, for each tuple returned by $\hat{\varphi}'$, 6^k tuples will be returned by $\hat{\varphi}$. But, we know that $\dot{\varphi}$ is a well-defined triangle query. This means that, for each $(3k)$ tuple of points $((\mathbf{a}_{1,1}, \mathbf{a}_{1,2}, \mathbf{a}_{1,3}), (\mathbf{a}_{2,1}, \mathbf{a}_{2,2}, \mathbf{a}_{2,3}), \ldots, (\mathbf{a}_{k,1}, \mathbf{a}_{k,2}, \mathbf{a}_{k,3}))$ satisfying $\dot{\varphi}$, also the tuples $((\mathbf{a}_{1,j_{1,1}}, \mathbf{a}_{1,j_{1,2}}, \mathbf{a}_{1,j_{1,3}}), (\mathbf{a}_{2,j_{2,1}}, \mathbf{a}_{2,j_{2,2}}, \mathbf{a}_{2,j_{2,3}}), \ldots, (\mathbf{a}_{k,j_{k,1}}, \mathbf{a}_{k,j_{k,2}}, \mathbf{a}_{k,j_{k,3}}))$, where $\sigma_i(1,2,3) = (j_{i,1}, j_{i,2}, j_{i,3})(1 \leq i \leq k; \sigma_i \in \mathcal{S}_3)$ and $\mathcal{S}_3$ is the set of all permutations of $\{1,2,3\}$, satisfy $\dot{\varphi}$. Therefor, $\hat{\varphi}$ and $\dot{\varphi}$ are equivalent according to definition 7.8.

We now combine the soundness and completeness lemmas, and use them to prove our main theorem for this section:

Theorem 7.19 (**Expressiveness of FO($\boldsymbol{\Delta}$)**)**.** Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\overline{R}_i$ be the corresponding constraint relation names of arity $6 \times ar(\hat{R}_i)$, for $1 \leq i \leq m$, and let $\overline{\sigma}$ be the spatial database schema $\{\overline{R}_1, \overline{R}_2, \ldots, \overline{R}_m\}$. The language FO($\boldsymbol{\Delta}$, $\hat{\sigma}$) is sound and complete for the affine-generic FO($+$, $\times$, $<$, 0, 1, $\overline{\sigma}$)-queries on triangle databases.

Proof. Let $\hat{\sigma} = \{\hat{R}_1, \hat{R}_2, \ldots, \hat{R}_m\}$ be a spatial triangle database schema. Let $\dot{R}_i$ be the corresponding spatial point relation names of arity $3 \times ar(\hat{R}_i)$, for $1 \leq i \leq m$, and let $\dot{\sigma}$ be the spatial database schema $\{\dot{R}_1, \dot{R}_2, \ldots, \dot{R}_m\}$. Let $\overline{R}_i (1 \leq i \leq m)$ be the corresponding constraint relation names of arity $6 \times ar(\hat{R}_i)$ and let $\overline{\sigma}$ be the spatial database schema $\{\overline{R}_1, \overline{R}_2, \ldots, \overline{R}_m\}$.

From Lemma 7.16 and Lemma 7.17, we can conclude that FO(Δ, $\hat{\sigma}$) is sound and complete for the FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$)-queries on triangle databases.

Gyssens, Van den Bussche and Van Gucht showed that FO($\{\mathbf{Between}^{(2)}\}$, $\dot{\sigma}$) is sound and complete for the affine-generic FO($+$, $\times$, $<$, 0, 1, $\overline{\sigma}$)-queries on geometric databases [43].

From the definition of triangle databases, we know that they are geometric databases. This concludes the proof. □

The following remark is important, we will come back to it at the end of this section.

Remark 7.20. In the proofs of Lemma 7.16 and Lemma 7.17, we only use the fact that triangles are convex objects having three corner points. We use no other properties of triangles.

The following corollary follows from the fact that FO({**Between**}, $\dot{\sigma}$) +**While** is sound and complete for the computable affine-generic queries on geometric databases [43]. The language FO($\mathbf{\Delta}$, $\hat{\sigma}$) + **While** is a language in which FO($\mathbf{\Delta}$, $\hat{\sigma}$)-definable relations can be created and which has a while-loop with FO($\mathbf{\Delta}$, $\hat{\sigma}$) -definable stop conditions.

Corollary 7.21 (**Expressiveness of** FO($\mathbf{\Delta}$, $\hat{\sigma}$)+ **While**)**.** Let $\hat{\sigma}$ be a spatial triangle database schema. The language FO($\mathbf{\Delta}$, $\hat{\sigma}$)+ **While** is sound and complete for the computable affine-generic queries on triangle databases.

We now give some examples of FO($\mathbf{\Delta}$, $\hat{\sigma}$)-queries. We illustrate some geometrical constructions in Example 7.22. Afterwards, we formulate queries on an example spatial triangle database in Example 7.23.

Example 7.22. We illustrate how to express that two triangles are similar, *i.e.*, each side of the first triangle is parallel to a side of the second triangle. We denote the formula expressing this by **Sim**.

We use the predicates **ColSeg** and **ParSeg**, expressing that two line segments are collinear and parallel respectively, to simplify the expression for **Sim**.

$$\begin{aligned}\mathbf{ColSeg}(\triangle_1, \triangle_2) := \mathbf{Seg}(\triangle_1) \wedge \mathbf{Seg}(\triangle_2) \wedge \\ \exists\, \triangle_3\ (\mathbf{Seg}(\triangle_3) \wedge \mathbf{PartOf}(\triangle_1, \triangle_3) \wedge \mathbf{PartOf}(\triangle_2, \triangle_3)).\end{aligned}$$

Here, $\mathbf{Seg}(\triangle_1)$ is a shorthand for

$$\begin{aligned}&\exists\, \triangle_4\ \exists\, \triangle_5\ (\mathbf{Point}(\triangle_4) \wedge \mathbf{Point}(\triangle_5) \wedge \\ &\qquad \forall\, \triangle_6\ ((\mathbf{Point}(\triangle_6) \wedge \mathbf{PartOf}(\triangle_6, \triangle_1)) \rightarrow (\mathbf{Between}_{\mathbf{\Delta}}(\triangle_4, \triangle_6, \triangle_5)))).\end{aligned}$$

The fact that two line segments are parallel is now defined as follows:

$$\begin{aligned}\mathbf{ParSeg}(\triangle_1, \triangle_2) := \mathbf{Seg}(\triangle_1) \wedge \mathbf{Seg}(\triangle_2) \wedge \forall\, \triangle_3\ \forall\, \triangle_4\ (\\ (\mathbf{ColSeg}(\triangle_1, \triangle_3) \wedge \mathbf{ColSeg}(\triangle_2, \triangle_4)) \rightarrow \\ \neg\exists\, \triangle_5\ (\mathbf{PartOf}(\triangle_5, \triangle_3) \wedge \mathbf{PartOf}(\triangle_5, \triangle_4))).\end{aligned}$$

Now we can write the expression for **Sim**:

$$\begin{aligned}&\mathbf{Sim}(\triangle_1, \triangle_2) := \\ &\exists\, \triangle_{1,1}\ \exists\, \triangle_{1,2}\ \exists\, \triangle_{1,3}\ \exists\, \triangle_{1,4}\ \exists\, \triangle_{1,5}\ \exists\, \triangle_{1,6}\ \exists\, \triangle_{2,1}\ \exists\, \triangle_{2,2}\ \exists\, \triangle_{2,3}\ \exists\, \triangle_{2,4}\ \exists\, \triangle_{2,5}\ \exists\, \triangle_{2,6}\ (\\ &\quad \bigwedge_{i=1}^{2} (\mathbf{CornerP}(\triangle_{i,1}, \triangle_{i,2}, \triangle_{i,3}, \triangle_i) \wedge \mathbf{CornerP}(\triangle_{i,1}, \triangle_{i,1}, \triangle_{i,2}, \triangle_{i,4}) \wedge \\ &\quad \mathbf{CornerP}(\triangle_{i,2}, \triangle_{i,2}, \triangle_{i,3}, \triangle_{i,5}) \wedge \mathbf{CornerP}(\triangle_{i,3}, \triangle_{i,3}, \triangle_{i,1}, \triangle_{i,6})) \wedge \\ &\quad \bigvee_{\sigma(1,2,3)=(i_1,i_2,i_3), \sigma\in\mathcal{S}_3} (\mathbf{ParSeg}(\triangle_{1,4}, \triangle_{2,(3+i_1)}) \wedge \mathbf{ParSeg}(\triangle_{1,5}, \triangle_{2,(3+i_2)}) \\ &\qquad\qquad \wedge \mathbf{ParSeg}(\triangle_{1,6}, \triangle_{2,(3+i_3)}))),\end{aligned}$$

where $\mathcal{S}_3$ is the set of all permutations of $\{1, 2, 3\}$.

□

We proceed with an example of a spatial database containing information about butterflies, and some FO($\mathbf{\Delta}$)-queries that can be asked to such a database.

Example 7.23. Consider a triangle database $\mathcal{D}$ over the schema $\hat{\sigma} = \{ButterflyB, PlantP, Rural\}$ that contains information about butterflies and flowers. The unary triangle relation $ButterflyB$ contains all regions where some butterfly B is spotted. The unary triangle relation $PlantP$ contains all regions where some specific plant P grows. We also have a unary triangle relation $Rural$, containing rural regions. It is known in biology that each butterfly appears close to some specific plant, as caterpillars only eat the leaves of their favorite plant. Suppose that it is also investigated that butterflies like to live in rural areas.

• Q_{10} : *Are all butterflies B spotted in regions where the plant P grows?* This query can be used to see if it is possible that a butterfly was spotted in a certain region. The query $Q_{10}()$ can be expressed by the formula

$$\neg(\exists\, \triangle_1\ \exists\, \triangle_2\ (ButterflyB(\triangle_1) \wedge \mathbf{RealTriangle}(\triangle_2) \wedge \mathbf{PartOf}(\triangle_2, \triangle_1) \wedge \neg(\exists\, \triangle_3\ (PlantP(\triangle_3) \wedge \mathbf{PartOf}(\triangle_2, \triangle_3))))).$$

Here, $\mathbf{RealTriangle}(\triangle)$ is a shorthand for $\neg\mathbf{Point}(\triangle) \wedge \neg\mathbf{Line}(\triangle)$.

• Q_{11} : *Give the region(s) where we have to search if we want to see butterfly B.* The query $Q_{11}(\triangle)$ can be expressed by the formula

$$\exists\, \triangle_2\ \exists\, \triangle_3\ (PlantP(\triangle_2) \wedge Rural(\triangle_3) \wedge \mathbf{PartOf}(\triangle, \triangle_2) \wedge \mathbf{PartOf}(\triangle, \triangle_3)).$$

• Q_{12} : *Give the region inside the convex hull of the search region for butterfly B.* It is much more convenient to search a convex region than having to deal with a very irregularly shaped region.

We first express how to test whether the region is convex (Q'_{12}), this will help understand the formula that computes the convex hull. The query $Q'_{12}()$ can be expressed by the formula

$$\forall\, \triangle_1\ \forall\, \triangle_2\ \forall\, \triangle_3\ \forall\, \triangle_4\ ((\bigwedge_{i=1}^{3} \mathbf{Point}(\triangle_i) \wedge \bigwedge_{i=1}^{3} Q_{11}(\triangle_i) \wedge \mathbf{CornerP}(\triangle_1, \triangle_2, \triangle_3, \triangle_4)) \Rightarrow (Q_{11}(\triangle_4))).$$

The expression

$$\exists\, \triangle_1\ \exists\, \triangle_2\ \exists\, \triangle_3\ \exists\, \triangle_4\ \exists\, \triangle_5\ \exists\, \triangle_6\ (\bigwedge_{i=1}^{3} \mathbf{Point}(\triangle_i) \wedge \bigwedge_{i=1}^{3} \mathbf{PartOf}(\triangle_i, \triangle_{i+3}) \wedge \bigwedge_{i=4}^{6} \mathbf{Q}_{11}(\triangle_i) \wedge \mathbf{CornerP}(\triangle_1, \triangle_2, \triangle_3, \triangle))$$

hence defines the query $Q_{12}(\triangle)$. For any three points in some triangles in Q_{11}, the triangle connecting them is added to Q_{12}. Figure 7.5 illustrates this. □

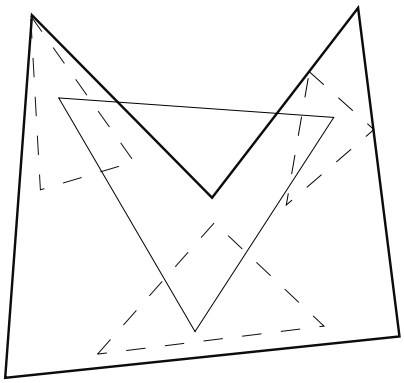


Figure 7.5: The convex hull of a set S of triangles is computed by adding all triangles constructed from three points that are inside three triangles of S.

Remark 7.24. The first two queries of Example 7.23 ask for relations between regions that can be expressed by the so-called 9-intersection model [24]. This model defines a relation between two regions by investigating the intersections between their boundaries, interiors and exteriors. As the boundary, interior and exterior of a region can be expressed in FO($+$, $\times$, $<$, 0, 1, $\overline{\sigma}$), and are affine invariant concepts [1], all relations that can be expressed by the 9-intersection model, can be expressed in FO($\mathbf{\Delta}$, $\hat{\sigma}$).

We now reconsider Remark 7.20. In the proofs of Lemma 7.16 and Lemma 7.17, we only used the fact that triangles are convex objects having three corner points. It is not difficult to prove that the predicate **PartOf** can be generalized to a predicate $\mathbf{PartOf}^{(n,k)}$, which arguments are n-dimensional convex objects with k corner points (*(n,k)-objects*) and that the language FO($\{\mathbf{PartOf}^{(n,k)}\}$) is sound and complete for the first-order affine-generic queries on (n, k)-objects.

In the context of this remark, we also want to refer to the work of Aiello and van Benthem [3, 4] on modal logics of space. They first propose a topological modal logic over regions, which can express "connectedness" and "parthood". By adding a "convexity" operator (expressed using a "betweenness" operator), they obtain an affine modal logic. Essentially, we do the same, as triangles are convex and connected sets, and we add the "parthood" operator **PartOf**.

In [4], the authors also motivate the use of finite unions of convex sets as basic elements for spatial reasoning. They argue that it is a very natural way for people to reason about objects. A fork, for example will be described as the union of its prongs and its handle.

[1] To be exact, they are topological concepts. The affinities of the plane are a subgroup of the homeomorphisms of the plane, so the invariance under the boundary and interior operations carry over naturally.

7.3.2 Safety of Triangle Database Queries

Triangle relations can represent infinite sets of triangles. In practice, however, spatial databases will contain only finite sets of triangles. The $ButterflyB$ and $Rural$ triangle relations of Example 7.23, for instance, will be modelled in practice using a finite number of triangles.

The question that arises naturally is whether the language FO($\mathbf{\Delta}$) returns a finite set of triangles when the input relations represent finite sets of triangles. The answer is "no" (see Example 7.25 below). In database theory this problem is usually referred to as the *safety problem*. Safety of FO($+$, $\times$, $<$, 0, 1)-queries is undecidable in general [7], so we cannot decide *a priori* whether a triangle database query will return a finite output or not.

The following example illustrates the fact that the language FO($\mathbf{\Delta}$) does not necessarily return finite output on finite input.

Example 7.25. Let $\hat{\sigma} = \{\hat{R}\}$ be a spatial triangle database schema, with $\hat{R}$ a triangle relation containing a finite number of triangles. Consider the following spatial triangle database queries:

• Q_{13} : *Give all triangles that are part of some triangle of* $\hat{R}$.
The query $Q_{13}(\triangle)$ is expressed in FO(Δ, $\hat{\sigma}$) by the formula

$$\exists\, \triangle'\ (\hat{R}(\triangle') \wedge \mathbf{PartOf}(\triangle, \triangle')).$$

• Q_{14} : **Give all triangles that intersect some triangle of** $\hat{R}$**.** The query $Q_{14}(\triangle)$ can be expressed by the formula

$$\exists\, \triangle'\ (\hat{R}(\triangle') \wedge \mathbf{Intersect}(\triangle, \triangle')).$$

• Q_{15} : **Give all the corner points of all triangles of** $\hat{R}$**.** The query $Q_{15}(\triangle)$ can be expressed by the formula

$$\begin{aligned}\exists\, \triangle_1\ \exists\, \triangle_{1,2}\ \exists\, \triangle_{1,3}\ (\hat{R}(\triangle_1) \wedge (&\mathbf{CornerP}(\triangle, \triangle_{1,2}, \triangle_{1,3}, \triangle_1)\\ &\vee \mathbf{CornerP}(\triangle_{1,2}, \triangle, \triangle_{1,3}, \triangle_1) \vee \mathbf{CornerP}(\triangle_{1,2}, \triangle_{1,3}, \triangle, \triangle_1))).\end{aligned}$$

The queries Q_{13} and Q_{14} return an infinite set of triangles. The query Q_{15} returns a finite number of triangles on the condition that the input relation $\hat{R}$ is finite. □

As we cannot decide whether a given triangle database query will return a finite result, we turn to the question of determining whether the result of the query is finite or not, after executing the query. The answer is affirmative:

Proposition 7.26 (**Finiteness of triangle relations is decidable**)**.** It is decidable whether a triangle relation consists of a finite number of triangles. Moreover, there exists a FO(Δ)$\{\hat{R}\}$ query that decides whether the triangle relation named $\hat{R}$ consists of a finite number of triangles.

Proof. A triangle relation of arity k corresponds to a semi-algebraic set in $\mathbb{R}^{6k}$. The canonical bijection $can \circ can_{tr} : \left((\mathbb{R}^2)^3\right)^k \to \mathbb{R}^{6k}$ establishes this correspondence. A

triangle relation is finite if and only if the corresponding semi-algebraic set contains a finite number of points (in $\mathbb{R}^{(6k)}$). It is well known that there exists a FO(+, ×, <, 0, 1)-formula deciding whether a semi-algebraic set contains a finite number of points. Also, the fact that a triangle relation contains a finite number of k-tuples of triangle is affine-invariant. From the fact that the property is affine-invariant and expressible in FO(+, ×, <, 0, 1), it follows (from Theorem 7.19) that there is a a FO(Δ, $\{\hat{R}\}$)-formula expressing whether a triangle relation $\hat{R}$ is finite or not. □

We now have a means of deciding whether a triangle relation is finite, but it seems this requirement is too restrictive.

In Definition 7.1 in Section 7.2, we introduced the concept *drawing* of a triangle. We now straightforwardly extend this definition to spatial triangle databases.

Definition 7.27 (**Drawing of a triangle relation**)**.** Let $\hat{R}$ be a triangle relation of arity one. The *drawing* of $\hat{R}$ is the two-dimensional figure that is the union of the drawings of all triangles in $\hat{R}$.

For the remainder of this text, we restrict triangle relations (and triangle database queries) to be unary. It is not clear immediately if it would make sense to define drawings on relations or queries with an arity greater than one. For example, consider a binary relation containing only one tuple of line-adjacent non-degenerated triangles. If we draw this relation, we would like to draw both triangles participating in the relation. This gives the same result as the drawing of a unary relation containing two tuples. So the drawing apparently "wipes out" the relationship between the triangles.

We also remark the following.

Remark 7.28. Different triangle relations can have the same drawing. Therefore, it seems natural to extend the strict notion of finiteness of a triangle relation to the existence of a finite triangle relation having the same drawing. Query Q_1 from Example 7.25, for instance, seems to be a query we would like to call "finite", because there exists a finite union of triangles with the same drawing. Indeed, the drawing of the union of all triangles that are part of a given triangle, is the same as the drawing of the given triangle itself. Query Q_2 clearly returns an infinite set of triangles that is cannot be represented as a finite union of triangles. This is the type of query we don't want to allow.

Fortunately, given the output of a unary query, we can determine whether its drawing can be represented as a finite union of triangles.

Proposition 7.29 (**Finite triangle representation**)**.** Let $\hat{\sigma}$ be a spatial triangle database schema. Given a unary triangle database query $\hat{Q}$ that is expressible in FO($\{$**PartOf**$\}$, $\hat{\sigma}$) and a spatial triangle database $\mathcal{D}$ over $\hat{\sigma}$, it is decidable whether the unary relation, named $\hat{R}_{\hat{Q}}$, containing $\hat{Q}(\mathcal{D})$ can be represented as a finite union of triangles. Furthermore, there exists a FO(Δ, $\hat{\sigma}'$)-formula deciding this for $\hat{\sigma}' = \hat{\sigma} \cup \{\hat{R}_{\hat{Q}}\}$.

Proof. It is clear that if the drawing of a triangle relation can be represented as a finite union of triangles, it can be represented by a FO(+, ×, <, 0, 1)-formula using only polynomials of degree at most one. A set that can be described using polynomials

of at most degree one, is called a semi-linear set. It is well-known that the bounded semi-linear sets are the same as finite unions of bounded polytopes (which triangles are).

So, if we can check whether the drawing of a (possibly infinite) set of triangles is bounded and can be represented using polynomials of degree at most one, we know that the set can be represented by a finite number of triangles.

Checking whether the drawing of a triangle relation $\hat{R}$ is bounded can be done easily in FO($\{$**PartOf**$\}$)$\{\hat{R}\}$. The following formula performs this check.

$$\textbf{IsBounded}() := \exists\, \triangle_1\ \forall\, \triangle_2\ \ (\hat{R}(\triangle_2) \rightarrow \textbf{PartOf}(\triangle_2, \triangle_1)).$$

Also, we can decide whether a two-dimensional[2] semi-algebraic set can be represented using polynomials of degree at most k, for any natural number k [52]. There exists a FO(+, $\times$, <, 0, 1)-formula deciding this [52]. It is clear that the drawing of a unary triangle relation is a semi-algebraic set.

From the facts that (*i*) computing the drawing of a triangle relation is an affine-generic query that can be expressed in FO(+, $\times$, <, 0, 1) and that (*ii*) checking whether a triangle relation has a bounded drawing can be expressed in FO($\{$**PartOf**$\}$, $\hat{\sigma}$) and that (*iii*) there exists a FO(+, $\times$, <, 0, 1)-formula deciding whether the drawing of a triangle relation can be expressed by polynomials of degree at most one can be done in FO(+, $\times$, <, 0, 1) and, finally, that (*iv*) the fact that the drawing of a triangle relation can be expressed by polynomials of degree at most one is affine-invariant, we conclude that we can decide whether a triangle relation has a finite representation, and that we can construct a FO($\{$**PartOf**$\}$, $\hat{\sigma}$)-formula deciding this. □

We now show that, if the drawing of the output of a triangle database query is representable as a finite set of triangles, we can compute such a finite triangle representation in FO($\{$**PartOf**$\}$, $\hat{\sigma}$).

In Chapter 6, we proposed an algorithm (Algorithm 3) that computes an affine invariant triangulation of a set of triangles. Recall that this algorithm computes the drawing of the input triangles, then partitions this drawing into a set of convex polygons according to the carriers of its boundary segments and finally triangulates convex polygons by connecting their center of mass to their corner points.

We assumed in Chapter 6 that the input set of triangles for Algorithm 3 was finite. On an infinite collection of triangles for which there exists a finite collection of triangles with the same drawing, this algorithm would work also correctly, however. The triangulation described by Algorithm 3 therefor seems a good candidate for representing infinite sets of triangles by finite sets of triangles. But, in [44], we conjectured that the triangulation described in Algorithm 3 cannot be expressed in FO($\{$**PartOf**$\}$, $\{\hat{R}\}$). The reason for this is the conjecture [49] that the center of mass of a polygon, which is an affine-invariant, cannot be expressed in FO(+, $\times$, <, 0, 1), and therefore, also not in FO($\{$**PartOf**$\}$, $\{\hat{R}\}$).

Conjecture 7.3.1 (Center of mass of a convex polygon not expressible in FO(+, $\times$, <, 0, 1)). [49] Let $P = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_k\}$ be a set of corner points that

[2]Note that this is not true for arbitrary dimensions.

represent a convex polygon. Assume that $k > 3$. The center of mass of the polygon represented by P cannot be expressed in FO(+, ×, <, 0, 1).

Remark that the center of mass of an arbitrary set of points is not expressible in FO(+, ×, <, 0, 1).

So, Algorithm 3 cannot be used. But, this algorithm computes a *partition* of the input into triangles, which is not a requirement here. If we relax the requirement of having a partition of the original figure into triangles down to having a *finite union* of (possibly overlapping) triangles representing the figure, we can avoid the computation of the center of mass. The adapted algorithm (Algorithm 8) is given below. It is described at an higher level than the description of Algorithm 3, as we are not interested in computational complexity here.

Algorithm 8 $AfTr(S)$.

Require: S is a unary triangle relation that can be represented as a finite union of triangles.

1: Compute the boundary B_S of S. B_S is a finite set of line segments and points.
2: Compute the set of carriers for all line segments of B_S. Those carriers partition S into a finite union of open convex polygons, points and open line segments. All closures of line segments that do not form a side of one of the convex polygons, together with all points that are not a corner point of one of the convex polygons are returned as degenerated triangles. Remark that we can return the closures of the line segments as S originally is a union of closed triangles, closed line segments and points.
3: **for** each polygon **do**
4: output the finite set of triangles that connect three distinct corner points of the polygon
5: **end for**

Given an unary triangle relation $\hat{R}$, we denote the result of Algorithm 8 on input $\hat{R}$ by the *affine finite triangle representation* of $\hat{R}$, or, abbreviated, $AfTr(\hat{R})$. Now we show that $AfTr(\hat{R})$ can be computed in FO(**Delta**, $\hat{R}$), provided that $\hat{R}$ can be represented as a finite union of triangles.

Proposition 7.30 (**Affine finite triangle representation**)**.** Given a unary triangle relation $\hat{R}$ that can be represented as a finite union of triangles, then there exists an FO($\mathbf{\Delta}$, $\{\hat{R}\}$)-formula returning $AfTr(\hat{R})$.

Proof. We use the fact that all affine-generic semi-algebraic queries on triangle databases can be expressed in FO(**Delta**, $\hat{R}$). Therefor, we have to prove that, first, the affine finite triangle representation is affine-invariant and, second, that the affine finite representation is expressible in FO(+, ×, <, 0, 1).

The affine finite representation is an affine invariant.

We only have to prove this for Step 3 of Algorithm 8. The rest is shown in Property 6.2.4 of Chapter 6.

Let $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_k\}$ be the set of corner points of a convex polygon P, where $k \geq 3$. Let α be an affinity of the plane. The set $\{\alpha(\mathbf{a}_1), \alpha(\mathbf{a}_2), \ldots, \alpha(\mathbf{a}_k)\}$ contains the corner

points of the convex polygon $\alpha(P)$. It is clear that, for each triangle $(\mathbf{a}_h, \mathbf{a}_i, \mathbf{a}_j)$ (such that $h \neq i$, $i \neq j$, $h \neq j$ and $1 \leq h, i, j \leq k$) connecting three corner points of P, the triangle $\alpha(\mathbf{a}_h, \mathbf{a}_i, \mathbf{a}_j) = (\alpha(\mathbf{a}_h), \alpha(\mathbf{a}_i), \alpha(\mathbf{a}_j))$ is an element of the set of triangles connecting three corner points of $\alpha(P)$.

The affine finite representation is computable in FO(+, ×, <, 0, 1).

In FO(+, ×, <, 0, 1), it is possible to compute the boundary of a semi-linear set (Line 1 of Algorithm 8). It is also possible to compute the carriers of all boundary line segments, and their intersection points (Line 2). It can be expressed that two points belong to the same convex polygon, namely, by expressing that the line segment in between them is not intersected by a carrier. Finally, the set of all triples of intersection points between carriers that belong to the same convex polygon can be computed in FO(+, ×, <, 0, 1) (Lines 3 through 5). From the fact that the triangle representation is affine invariant and computable in FO(+, ×, <, 0, 1), it follows that it is computable in FO($\mathbf{\Delta}$). □

This section on safety finishes the "spatial" part of this chapter. In the remaining part, we develop a query language for spatio-temporal triangle databases.

7.4 Spatio-temporal Triangle Queries

In this section, we will extend the spatial triangle logic FO({**PartOf**}) to a logic over spatio-temporal triangles, *i.e.*, triples of co-temporal points in $(\mathbb{R}^2 \times \mathbb{R})$. The genericity classes we consider in this section, are the group $(\mathcal{A}_{st}, \mathcal{A}_t)$ of time-dependent affinities, the group $(\mathcal{V}_{st}, \mathcal{A}_t)$ of velocity-preserving transformations and the group $(\mathcal{AC}_{st}, \mathcal{A}_t)$ of acceleration-preserving transformations. The first group is a natural spatio-temporal extension of the affinities of space. We also include the two other groups, because they are very relevant from a practical point of view, and because the point languages we previously identified as generic for those groups were not very intuitive.

Recall that $\mathcal{A}_t$ is the group of the affinities on the time line and that the elements of $\mathcal{A}_{st}$ are of the form

$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ t \end{pmatrix} \mapsto \begin{pmatrix} \alpha_{11}(t) & \alpha_{12}(t) & \cdots & \alpha_{1n}(t) \\ \alpha_{21}(t) & \alpha_{22}(t) & \cdots & \alpha_{2n}(t) \\ \vdots & \vdots & \cdots & \vdots \\ \alpha_{n1}(t) & \alpha_{n2}(t) & \cdots & \alpha_{nn}(t) \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} \beta_1(t) \\ \beta_2(t) \\ \vdots \\ \beta_n(t) \end{pmatrix},$$

where the matrix of the $\alpha_{ij}(t)$ is an affinity for each value of t. The group $(\mathcal{AC}_{st}, \mathcal{A}_t)$ is the subgroup of $(\mathcal{A}_{st}, \mathcal{A}_t)$ in which the functions α_{ij} are constants and the functions β_{ij} are linear functions of time. The group $(\mathcal{V}_{st}, \mathcal{A}_t)$ is the subgroup of $(\mathcal{AC}_{st}, \mathcal{A}_t)$ where the β_{ij} are constants too.

In Chapter 4, we proposed point languages capturing exactly those genericity classes. Table 7.1 summarizes the point languages expressing all $(\mathcal{F}_{st}, \mathcal{F}_t)$-generic queries, for the above groups $(\mathcal{F}_{st}, \mathcal{F}_t)$. As we will always assume, in this section, that

the underlying dimension is 2, we adapted the table accordingly. Now we propose spatio-temporal point languages that have the same expressivity as the languages listed in Table 7.1, but on spatio-temporal triangle databases.

$(\mathcal{F}_{st}, \mathcal{T}_t)$	Set of point predicates $\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$
$(\mathcal{A}_{st}, \mathcal{A}_t)$	$\{\mathbf{Between}^{\text{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$
$(\mathcal{AC}_{st}, \mathcal{A}_t)$	$\{\mathbf{Between}^{(2+1)}, \mathbf{Before}\}$
$(\mathcal{V}_{st}, \mathcal{A}_t)$	$\{\mathbf{Between}^{(2+1)}, \mathbf{Before}, \mathbf{EqSpace}\}$

Table 7.1: The point logics FO($\Pi(\mathcal{F}_{st}, \mathcal{T}_t)$) capturing the FO $(\mathcal{F}_{st}, \mathcal{T}_t)$-generic queries, for the classes $(\mathcal{A}_{st}, \mathcal{A}_t)$, $(\mathcal{AC}_{st}, \mathcal{A}_t)$ and $(\mathcal{V}_{st}, \mathcal{A}_t)$.

We will start with the most general transformation group, the group $(\mathcal{A}_{st}, \mathcal{A}_t)$ of time-dependent affinities.

7.4.1 Predicates Invariant under Time-dependent Affinities

In this section, we propose a set of spatio-temporal triangle predicates such that the spatio-temporal triangle logic with this predicate set, captures exactly the $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic queries on spatio-temporal triangle databases that are expressible in FO(+, $\times$, $<$, 0, 1). We can prove this by comparing the expressiveness of this spatio-temporal triangle logic with the language FO($\{\mathbf{Between}^{\text{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$), when used as a spatio-temporal triangle query language (see Definition 7.9). Recall also that we will have to make sure that the result of a spatio-temporal triangle query is a consistent spatio-temporal triangle relation.

The nature of the class $(\mathcal{A}_{st}, \mathcal{A}_t)$ is such that $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic queries can describe snapshots of a spatio-temporal database in fairly much detail, *i.e.*, all affine-invariant properties of the snapshot can be expressed. In between snapshots, the expressive power of $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic queries is more limited. This follows directly from the fact that an element of $(\mathcal{A}_{st}, \mathcal{A}_t)$ transforms each snapshot with another affinity. We now want to construct a $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic query language for spatio-temporal triangle databases. This means we will be able to describe a spatio-temporal triangle database by means of its snapshots, which are collections of snapshots of spatio-temporal triangles in $(\mathbb{R}^2 \times \{\tau_0\})^3$, for some $\tau_0 \in \mathbb{R}$. The basic objects for our new language will be, accordingly, triples of co-temporal points. In this section, we will call these triples of points *triangle snapshots*. Triangle snapshot variables will be denoted $\triangle^{st}, \triangle_1^{st}, \triangle_2^{st}, \ldots$ and triangle snapshot constants by $T^{st}, T_1^{st}, T_2^{st}, \ldots$. If we want to emphasize the connection between a triangle snapshot and its corner points, we use the notation T_{pqr}^{st}.

In our search for a set of predicates on triangle snapshots for a $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic query language, or, a language with the same expressive power as the language FO($\{\mathbf{Between}^{\text{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$) on spatio-temporal triangle databases, the following observations are helpful.

(i) In Chapter 4, we showed that we need the binary predicate **Before** on points to

reflect the monotonicity of time, which is preserved by the transformation group $(\mathcal{A}_{st}, \mathcal{A}_t)$.

(ii) The predicate $\mathbf{Between}^{\text{Cotemp}}$ is used to express affine-invariant properties of co-temporal points.

(iii) In Section 7.3, we showed that the predicate **PartOf** has the same expressive power as the predicate **Between**, on (spatial) triangles.

From observation (i) it follows that the query language we want to construct should be able to express the order on triangle snapshots. We introduce the triangle snapshot predicate $\mathbf{Before}_{\triangle}$, which, when applied to two triangle snapshots, expresses that the first one is strictly before or co-temporal with the second one. We will define this more formally later.

From observation (ii) and (iii), we conclude that we can use, slightly adapted, the predicate **PartOf** on co-temporal triangle snapshots, we will denote it $\mathbf{PartOf}^{\text{Cotemp}}$. This will allow us to express snapshots of spatio-temporal triangle databases in an affine-invariant way. Concluding, the set of spatio-temporal triangle predicates we are looking for should contain the elements $\mathbf{PartOf}^{\text{Cotemp}}$ and $\mathbf{Before}_{\triangle}$. Because, in the end, we want to express al queries expressible in FO($\{\mathbf{Between}^{\text{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$), on spatio-temporal triangle databases, we still have to look for a (set of) triangle snapshot predicate(s) capturing the expressive power of the predicate $\mathbf{EqCr}^{ST}$.

We repeat the definition of the point predicate $\mathbf{EqCr}^{ST}$. For six spatio-temporal points $p_1, p_2, p_3, q_1, q_2, q_3 \in ((\mathbb{R}^2 \times \mathbb{R}))$, $\mathbf{EqCr}^{ST}(p_1, p_2, p_3, q_1, q_2, q_3)$ expresses that the cross-ratio of the three co-temporal and collinear points p_1, p_2 and p_3 equals the cross-ratio of the time coordinates τ_{q_1}, τ_{q_2} and τ_{q_3} of the points q_1, q_2 and q_3. The expression $\mathbf{EqCr}^{ST}(p_1, p_2, p_3, q_1, q_2, q_3)$ implicitly refers to a movement. Indeed, the line segment defined by the points p_1 and p_3 and the interval $[\tau_{q_1}, \tau_{q_3}]$ can be interpreted as the spatial and temporal projection of a linear movement with constant speed and we can then interpret $\mathbf{EqCr}^{ST}(p_1, p_2, p_3, q_1, q_2, q_3)$ as an expression of the fact that when an object moves with constant speed from p_1 to p_3 during the interval $[\tau_{q_1}, \tau_{q_3}]$, it passes p_2 at time moment τ_{q_2}.

There is one obvious way to define the speed of a moving point. For moving triangles, or moving objects in general, the definition of *speed* is somewhat ambiguous. Triangles can move by changing their position, but also by changing their shape. We define the speed (resp., acceleration) of a moving triangle as the speed (resp., acceleration) of is moving *center of mass*. Hence, a triangle that is growing or shrinking, but its center of mass remains in the same position, has zero speed. Based on that definition, we propose a spatio-temporal triangle database query language, with the triangle predicates $\mathbf{PartOf}^{\text{Cotemp}}$, $\mathbf{Before}_{\triangle}$ and **Cas** (which is an abbreviation of "Constant Average Speed"). The predicate **Cas** takes six arguments $\triangle_1^{st}, \triangle_2^{st}, \ldots, \triangle_6^{st}$. The first three triangle snapshots, $\triangle_1^{st}, \triangle_2^{st}$ and $\triangle_3^{st}$, are co-temporal and their barycenters are collinear. The last three triangle snapshots, $\triangle_4^{st}, \triangle_5^{st}$ and $\triangle_6^{st}$, indicate three different time moments. Furthermore, the cross-ratio of the barycenters of $\triangle_1^{st}, \triangle_2^{st}$ and $\triangle_3^{st}$ is the same as the cross-ratio of the time coordinates of $\triangle_4^{st}, \triangle_5^{st}$ and $\triangle_6^{st}$. Intuitively, this predicate, similar to the point predicate $\mathbf{EqCr}^{ST}$, approximates or estimates a

linear movement. Given the time interval during which a triangle moves from the first position to the second one, it estimates, assuming the triangle moves with constant speed, how long it will take to reach the position of the third triangle.

It turns out, however, that the language with these three triangle predicates is not very intuitive to express properties of the shape of triangles, e.g., their relative areas. Therefor, we will also propose an alternative language. This language has exactly the same expressivity as the first one, but offers a more direct means to express shape properties of triangles. We propose to replace the predicate **Cas** by the predicate **Lex** (which is an abbreviation for "Linear Expansion"). This predicate also takes six arguments $\triangle_1^{st}$, $\triangle_2^{st}$, ..., $\triangle_6^{st}$. The first three triangle snapshots, $\triangle_1^{st}, \triangle_2^{st}$ and $\triangle_3^{st}$, are co-temporal and both $\mathbf{PartOf}^{\text{Cotemp}}(\triangle_1^{st}, \triangle_2^{st})$ and $\mathbf{PartOf}^{\text{Cotemp}}(\triangle_2^{st}, \triangle_3^{st})$ hold. The other three triangle snapshots exist at three different time moments. Finally, the cross-ratio of the time coordinates of $\triangle_4^{st}, \triangle_5^{st}$ and $\triangle_6^{st}$ equals the cross ratio of the areas of the three first triangles. More exactly,

$$\frac{|A(\triangle_2^{st}) - A(\triangle_1^{st})|}{|A(\triangle_3^{st}) - A(\triangle_1^{st})|} = \frac{|\tau_{\triangle_2^{st}} - \tau_{\triangle_1^{st}}|}{|\tau_{\triangle_3^{st}} - \tau_{\triangle_1^{st}}|},$$

where $\tau_{\triangle_i^{st}}$ denotes the time moment at which $\triangle_i^{st}$ exists and $A(\triangle_i^{st})$ denotes the area of the triangle $\triangle_i^{st}$. Intuitively, this predicate approximates or estimates a linear growth or expansion. Given the time interval during which the first triangle expanded into the second one, it estimates, assuming the triangle grows linearly, how long it will take to reach the area of the third triangle.

In applications where objects are not growing or shrinking, a language with the predicate **Cas** may be preferred, whereas in applications where objects do change their shape, the predicate **Lex** may be preferred. Of course, one can also include both predicates to make the language suitable for all types of applications.

We will prove that the both the languages FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$) and FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Lex}\}$) are sound and complete for the $(\mathcal{A}_{st}, \mathcal{A}_t)$-generic first-order spatio-temporal database queries.

7.4.1.1 Expressiveness of the Language FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$)

In this section, we first give the definitions of the triangle predicates $\mathbf{PartOf}^{\text{Cotemp}}$, $\mathbf{Before}_{\triangle}$ and **Cas**. Next, we show that the language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}$, **Cas**$\}$) produces queries that are well-defined on spatio-temporal triangle databases. After that, we show its expressive power. We end with some examples.

Definition 7.31 (**The triangle snapshot predicate $\mathbf{PartOf}^{\mathbf{Cotemp}}$**). Let $T_1^{st} = (p_{1,1}, p_{1,2}, p_{1,3})$ and $T_2^{st} = (p_{2,1}, p_{2,2}, p_{2,3})$ be two triangle snapshots. The binary predicate $\mathbf{PartOf}^{\text{Cotemp}}$, applied to T_1^{st} and T_2^{st} expresses that $p_{1,1}$, $p_{1,2}$ and $p_{1,3}$ (resp., $p_{2,1}$, $p_{2,2}$ and $p_{2,3}$) are co-temporal and that the convex closure of the three points $p_{1,1}$, $p_{1,2}$ and $p_{1,3}$ is a subset of the convex closure of the three points $p_{2,1}$, $p_{2,2}$ and $p_{2,3}$.

Definition 7.32 (**The triangle snapshot predicate $\mathbf{Before}_{\triangle}$**). Let $T_1^{st} = (p_{1,1}$, $p_{1,2}, p_{1,3})$ and $T_2^{st} = (p_{2,1}, p_{2,2}, p_{2,3})$ be two triangle snapshots. The binary predicate $\mathbf{Before}_{\triangle}$, applied to T_1^{st} and T_2^{st} expresses that $p_{1,1}$, $p_{1,2}$ and $p_{1,3}$ (resp., $p_{2,1}$, $p_{2,2}$

and $p_{2,3}$) are co-temporal and that the time coordinate $\tau_{p_{1,1}}$ of $p_{1,1}$ is smaller than or equal to the time coordinate $\tau_{p_{2,1}}$ of $p_{2,1}$.

Definition 7.33 (**The triangle snapshot predicate Cas**)**.** Let $T_1^{st} = (p_{1,1}, p_{1,2}, p_{1,3}), T_2^{st} = (p_{2,1}, p_{2,2}, p_{2,3}), \ldots, T_6^{st} = (p_{6,1}, p_{6,2}, p_{6,3})$ be six triangle snapshots. Let q_1 (resp., q_2, q_3) be the barycenter of T_1^{st} (resp., T_2^{st}, T_3^{st}). The 6-ary predicate **Cas**, applied to T_1^{st}, T_2^{st}, ..., T_6^{st} expresses that $p_{i,1}$, $p_{i,2}$ and $p_{i,3}$ are co-temporal for $i = 1 \ldots 6$, that q_1, q_2 and q_3 are collinear and that the cross-ratio of the points q_1, q_2 and q_3 is the same as the cross-ratio of the time coordinates $\tau_{p_{4,1}}$, $\tau_{p_{5,1}}$ and $\tau_{p_{6,1}}$ of $p_{4,1}$, $p_{5,1}$ and $p_{6,1}$, respectively.

We now show, by induction on their structure, that the FO($\{\mathbf{PartOf}^{\text{Cotemp}}$, $\mathbf{Before}_{\triangle}, \mathbf{Cas}\}$)-queries are well-defined on spatio-temporal triangle databases.

Lemma 7.34 (FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$) **is well-defined**)**.** Let $\hat{\sigma}^{st} = \{\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}\}$ be a spatio-temporal triangle database schema. Let $\mathcal{D}^{st}$ be a consistent spatio-temporal triangle database over $\hat{\sigma}^{st}$. For each FO($\{\mathbf{PartOf}^{\text{Cotemp}}$, $\mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}^{st}$)-query $\hat{Q}$, $\hat{Q}(\mathcal{D}^{st})$ is a consistent triangle relation.

Proof. Let $\hat{\sigma}^{st} = \{\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}\}$ be a spatio-temporal triangle database schema. Let $\mathcal{D}^{st}$ be a consistent spatial triangle database over $\hat{\sigma}^{st}$.

We prove this lemma by induction on the structure of FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}$, $\mathbf{Cas}\}$, $\hat{\sigma}$)-queries. The atomic formulas of FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}$) are equality expressions on spatio-temporal triangle variables, expressions of the form $\mathbf{PartOf}^{\text{Cotemp}}(\triangle_1^{st}, \triangle_2^{st})$, expressions of the form $\mathbf{Before}_{\triangle}(\triangle_1^{st}, \triangle_2^{st})$, expressions of the form $\mathbf{Cas}(\triangle_1^{st}, \triangle_2^{st}, \ldots, \triangle_6^{st})$, and expressions of the form $\hat{R}_i^{st}(\triangle_1^{st}, \triangle_2^{st}, \ldots, \triangle_{ar(\hat{R}_i)}^{st})$, where $\hat{R}_i^{st} (1 \leq i \leq m)$ is a relation name from $\hat{\sigma}^{st}$. More complex formulas can be constructed using the Boolean operators $\wedge$, $\vee$ and $\neg$ and existential quantification.

For the atomic formulas, it is easy to see that, if two triangles T_1^{st} and T_2^{st} satisfy the conditions $T_1^{st} =_{\triangle} T_2^{st}$, $\mathbf{PartOf}^{\text{Cotemp}}(T_1^{st}, T_2^{st})$, or $\mathbf{Before}_{\triangle}(T_1^{st}, T_2^{st})$ that also $T_3^{st} =_{\triangle} T_4^{st}$ respectively $\mathbf{PartOf}^{\text{Cotemp}}(T_3^{st}, T_4^{st})$, $\mathbf{Before}_{\triangle}(T_3^{st}, T_4^{st})$ are true iff $T_1 =_{\triangle} T_3$ and $T_2 =_{\triangle} T_4$ are true. As we assume the input database $\mathcal{D}$ to be consistent, the atomic formulas of the type $\hat{R}_i^{st}(\triangle_1^{st}, \triangle_2^{st}, \ldots, \triangle_{ar(\hat{R}_i^{st})}^{st})$, where $(1 \leq i \leq m)$, trivially return consistent triangle relations.

For the predicate **Cas**, the proof is less straightforward. First, it is true that any pair of triangles T^{st} and $T^{st'}$ such that $T^{st} =_{\triangle} T^{st'}$ have the same center of mass. Note that this center of mass, which is represented by a degenerated triangle, only has one representation. Second, all corner points representing a spatio-temporal triangle are co-temporal. Therefor, we can conclude that the cross-ratio of the time coordinates of three triangles T_1^{st}, T_2^{st} and T_3^{st} is the same as the cross-ratio of the time coordinates of any triple of triangles $T_1^{st'}$, $T_2^{st'}$ and $T_3^{st'}$, such that $T_l^{st} =_{\triangle} T_l^{st'} (1 \leq l \leq 3)$. It now follows from the first and second statements, that given the spatio-temporal triangles T_1^{st}, T_2^{st}, T_3^{st}, T_4^{st}, T_5^{st} and T_6^{st},

$$\mathbf{Cas}(T_1^{st}, T_2^{st}, T_3^{st}, T_4^{st}, T_5^{st}, T_6^{st}) \leftrightarrow \mathbf{Cas}(T_1^{st'}, T_2^{st'}, T_3^{st'}, T_4^{st'}, T_5^{st'}, T_6^{st'}),$$

for any $T_l^{st'}$ such that $T_l^{st} =_{\triangle} T_l^{st'} (1 \leq l \leq 6)$.

Now we have to prove that the composed formulas always return consistent triangle relations. Let $\hat{\varphi}$ and $\hat{\psi}$ be two formulas in FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}^{st}$), of arity k_φ and k_ψ respectively, already defining consistent triangle relations. Then, the formula $(\hat{\varphi} \wedge \hat{\psi})$ (resp., $(\hat{\varphi} \vee \hat{\psi})$) also defines a triangle relation. This follows from the fact that the free variables of $(\hat{\varphi} \wedge \hat{\psi})$ (resp., $(\hat{\varphi} \vee \hat{\psi})$) are free variables in $\hat{\varphi}$ or $\hat{\psi}$. The universe of all triangles is trivially consistent. If a consistent subset is removed from this universe, the remaining part is still consistent. Therefor, $\neg\hat{\varphi}$ is well-defined. Finally, because consistency is defined argument-wise, the projection $\exists T_1^{st}\, \hat{\varphi}(T_1^{st}, T_2^{st}, \ldots, T_{k_\varphi}^{st})$ is consistent. □

Theorem 7.35 (**Expressiveness of the language** FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$)). Let $\hat{\sigma}^{st}$ be a database schema. The language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}^{st}$) is sound and complete for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO-queries on spatio-temporal triangle databases.

As usual, we prove this theorem using the following two lemma's:

Lemma 7.36 (**Soundness of the language** FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$) **with respect to** FO($\{\mathbf{Between}^{\mathbf{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$)). Let $\hat{\sigma}^{st}$ be a spatio-temporal triangle database schema. The language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}^{st}$) is sound for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO-queries on spatio-temporal triangle databases.

Proof.
Let $\hat{\sigma}^{st} = \{\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}\}$ be a spatio-temporal triangle database schema. Similar to the proof of Lemma 7.16, this proof consists of two parts.

First, let $\dot{\sigma}^{st} = \{\dot{R}_1^{st}, \dot{R}_2^{st}, \ldots, \dot{R}_m^{st}\}$ be a spatio-temporal point database schema where the arity of $\dot{R}_i^{st}$ is $3 \times ar(\hat{R}_i^{st})$, for $i = 1, 2, \ldots, m$. We show that each formula of FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}^{st}$) can be translated in FO($\{\mathbf{Between}^{\text{Cotemp}}$, $\mathbf{Before}, \mathbf{EqCr}^{ST}\}$, $\dot{\sigma}^{st}$) . We to this by induction on FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}$, $\mathbf{Cas}\}$, $\hat{\sigma}^{st}$)-formulas. Next, we have to prove that each FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}$, $\mathbf{Cas}\}$, $\hat{\sigma}^{st}$)-query defines a consistent spatio-temporal triangle relation.

We start with the first part of this proof. Let $\dot{R}_i^{st} (1 \leq i \leq m)$ be the corresponding spatio-temporal point relation names of arity $3 \times ar(\hat{R}_i^{st})$ and let $\dot{\sigma}^{st}$ be the spatio-temporal (point) database schema $\{\dot{R}_1^{st}, \dot{R}_2^{st}, \ldots, \dot{R}_m^{st}\}$. Let $\hat{\varphi}$ be a FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}, \hat{\sigma}^{st}\}$)-formula.

Each triangle variable $\triangle^{st}$ in $\hat{\varphi}$ is translated naturally by three spatio-temporal point variables u_1, u_2, u_3. As we assume that all points composing a spatio-temporal triangle are co-temporal, we add the formula

$$\mathbf{Cotemp}(u_1, u_2) \wedge \mathbf{Cotemp}(u_2, u_3)$$

to the beginning of the translation of the sub-formula where $\triangle^{st}$ appears first. In the remainder of this proof we will omit these temporal constraints to keep formulas shorter and hence more readable, but always assume them.

The formulas in FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}^{st}$) are build from atomic formulas, composed by the operators $\wedge$, $\wedge$ and $\neq$ and quantification. The atomic

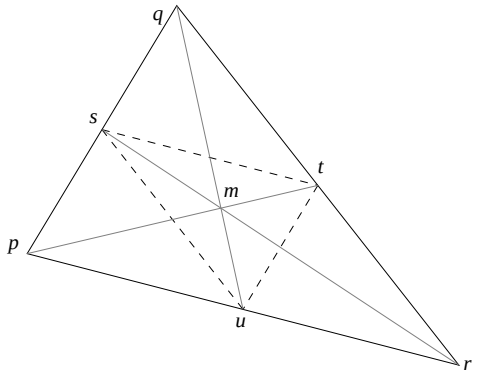


Figure 7.6: The center of mass of the triangle pqr is the intersection of the medians pt, qu and rs. Also, the lines tu, us and st are parallel to pq, qr and rp, respectively.

formulas of FO($\{\mathbf{PartOf}^{\mathrm{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}, \hat{\sigma}^{st}$) are equality constraints between spatio-temporal triangle variables, the triangle predicates $\mathbf{PartOf}^{\mathrm{Cotemp}}$, $\mathbf{Before}_{\triangle}$ and **Cas** applied to spatio-temporal triangle variables, and predicates of the form $\hat{R}_i^{st}(\triangle_1^{st}, \triangle_2^{st}, \ldots, \triangle_{ar(\hat{R}_i^{st})}^{st})(1 \leq i \leq m)$, where $\hat{R}_i^{st} \in \hat{\sigma}^{st}$. As this proof is analogous to the proof of Lemma 7.16, we only give the translation of the atomic formulas:

(i) The translation of $(\triangle_1^{st}{=}\triangle_2^{st})$ is

$$\bigvee_{\sigma(1,2,3)=(i_1,i_2,i_3),\sigma\in\mathcal{S}_3} (u_{1,1} = u_{2,i_1} \wedge u_{1,2} = u_{2,i_2} \wedge u_{1,3} = u_{2,i_3}),$$

where $\mathcal{S}_3$ is the set of all permutations of $\{1, 2, 3\}$.

(ii) In the proof of Lemma 7.16, we already showed that the predicate **PartOf** can be expressed in FO($\{$**Between**$\}$).

(iii) Expressions of the form $\mathbf{Before}_{\triangle}(\triangle_1^{st}, \triangle_2^{st})$ are translated as follows:

$$\mathbf{Before}(u_{1,1}, u_{2,1}).$$

Recall that the formulas expressing that the corner points of each triangle should be co-temporal are already added to the translation.

(iv) For the predicate **Cas**, first we need to to express in FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}, \dot{\sigma}^{st}$) that some point (in $(\mathbb{R}^2 \times \mathbb{R})$) is the center of mass of a triangle, represented by three other points, all co-temporal with the first point.

Figure 7.6 illustrates the construction of the center of mass of a triangle. Given a triangle T_{pqr}^{st}. There is only one way of constructing a triangle T_{stu}^{st} inscribed in T_{pqr}^{st} such that each side of T_{stu}^{st} is parallel to a side of T_{pqr}^{st}. The corner points of T_{stu}^{st} are in the middle of the sides of T_{pqr}^{st}. Hence, the center of mass of T_{pqr}^{st} is the intersection of the line segments connecting the corner points of stu with

the opposite corner point of T^{st}_{pqr}. The next formula expresses the predicate **CenterOM** in the language FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$). The free variables are v (representing the center of mass), u_1, u_2 and u_3 (representing the corner points of the triangle).

$$\begin{aligned}\exists w_1\,\exists w_2\,\exists w_3\,(&\mathbf{Between}^{\mathrm{Cotemp}}(u_1,w_1,u_2)\wedge\mathbf{Between}^{\mathrm{Cotemp}}(u_2,w_2,u_3)\wedge\\ &\mathbf{Between}^{\mathrm{Cotemp}}(u_3,w_3,u_1)\wedge\mathbf{Par}(u_1,u_2,w_2,w_3)\wedge\mathbf{Par}(u_2,u_3,w_1,w_3)\wedge\\ &\mathbf{Par}(u_3,u_1,w_1,w_2)\wedge\mathbf{Between}^{\mathrm{Cotemp}}(u_1,v,w_2)\wedge\\ &\mathbf{Between}^{\mathrm{Cotemp}}(u_2,v,w_3)\wedge\mathbf{Between}^{\mathrm{Cotemp}}(u_3,v,w_1)).\end{aligned}$$

Here, $\mathbf{Par}(v_1,v_2,v_3,v_4)$ is an abbreviation for the sub-formula

$$\neg\exists w\,(\mathbf{Collinear}(w,v_1,v_2)\wedge\mathbf{Collinear}(w,v_3,v_4)).$$

We now give the expression translating $\mathbf{Cas}(\triangle^{st}_1,\triangle^{st}_2,\triangle^{st}_3,\triangle^{st}_4,\triangle^{st}_5,\triangle^{st}_6)$. The following formula has (6×3) free point variables $u_{1,1}u_{1,2},u_{1,3},u_{2,1},u_{2,2},u_{2,3},\ldots,u_{6,1},u_{6,2},u_{6,3}$ that are the translation of the triangle variables $\triangle^{st}_1,\triangle^{st}_2,\ldots,\triangle^{st}_6$.

$$\begin{aligned}&\exists v_1\,\exists v_2\,\exists v_3\,(\\ &\bigwedge_{i=1}^{3}\mathbf{CenterOM}(v_i,u_{i,1},u_{i,2},u_{i,3})\wedge\mathbf{EqCr}^{ST}(v_1,v_2,v_3,u_{4,1},u_{5,1},u_{6,1})).\end{aligned}$$

(v) The translation of a formula of the type $\hat{R}^{st}(\triangle^{st}_1,\triangle^{st}_2,\ldots,\triangle^{st}_k)$, where $\hat{R}^{st}\in\hat{\sigma}^{st}$ is

$$\dot{R}(u_{1,1},u_{1,2},u_{1,3},u_{2,1},u_{2,2},u_{2,3},\ldots,u_{k,1},u_{k,2},u_{k,3}).$$

The correctness of this translation follows from Definition 7.2 and Remark 7.3. □

We can also show the possibility of the translation in the other direction. As the proof of Lemma 7.37 is completely analogous to the proof of Lemma 7.17, we omit it. The only new items are the translations of the spatio-temporal point predicates **Before** and $\mathbf{EqCr}^{ST}$ into FO($\{\mathbf{PartOf}^{\mathrm{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$). It is easy to see that these translations involve only replacing point variables by triangle variables that represent points.

Lemma 7.37 (**Completeness of** FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$))**.** Let $\hat{\sigma}^{st}$ be a spatio-temporal triangle database schema. The language FO($\{\mathbf{PartOf}^{\mathrm{Cotemp}}$, $\mathbf{Before}_{\triangle}, \mathbf{Cas}\}$, $\hat{\sigma}^{st}$) is complete for the $(\mathcal{AC}_{st},\mathcal{A}_t)$-generic FO-queries on spatio-temporal triangle databases.

We now propose an alternative language, with the same expressiveness as the language FO($\{\mathbf{PartOf}^{\mathrm{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Cas}\}$), which allows us to talk about areas of triangles.

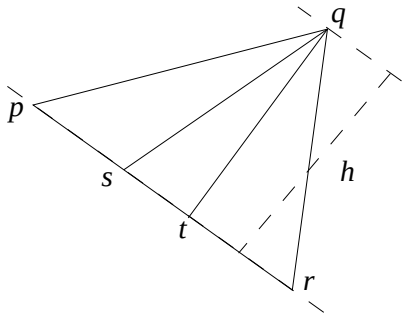


Figure 7.7: The area of T_{pqr} is to the area of T_{pqs} as the length of pr to the length of ps.

7.4.1.2 Expressiveness of the Language FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Lex}\}$)

We start this subsection with some geometric constructions. We will use those to express the predicate **Lex** in the language FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$). For these constructions, we assume that all spatio-temporal points and triangles are co-temporal.

Observation 7.4.1. Let two triangles T^{st}_{pqr} and T^{st}_{pqs} be given. If the point s is chosen on the line segment pr such that the cross ration of p, s and r equals c, then the areas of T^{st}_{pqr} and T^{st}_{pqs} have a ratio which is also equal to c. The correctness of this construction is easy to verify because the area of a triangle is half the length of its base line multiplied by its height. As T^{st}_{pqr} and T^{st}_{pqs} have both height h, their areas have the same relation as the lengths of their base lines ps and pq. Figure 7.7 illustrates this observation.

Suppose we have three triangles T^{st}_{pqr}, T^{st}_{pqs} and T^{st}_{pqt}, such that the points q,r, s and t are all collinear (suppose they are arranged as in Figure 7.7). Then it is true that

$$\frac{A(T^{st}_{pqt}) - A(T^{st}_{pqs})}{A(T^{st}_{pqr}) - A(T^{st}_{pqs})} = \frac{|\ st\ |}{|\ sr\ |}.$$

So it turns out to be possible to convert area ratios to cross-ratios of collinear points, for triangles that have the special configuration as described in Observation 7.4.1. We will observe next that it is possible, given three triangles T^{st}_1, T^{st}_2 and T^{st}_3 such that T^{st}_1 is part of T^{st}_2 and T^{st}_2 part of T^{st}_3, to construct triangles T^{st}_4 and T^{st}_5 with the same area as T^{st}_1 and T^{st}_2, respectively, such that T^{st}_4, T^{st}_5 and T^{st}_3 have this special configuration.

Observation 7.4.2. Given a pair of triangles T^{st}_1 and T^{st}_2 such that T^{st}_1 is part of T^{st}_2. Following the construction steps described below, we can construct a triangle T^{st}_3, with the same area as T^{st}_1. The triangle T^{st}_3 shares one side with T^{st}_2 and its third corner point is on one of the other sides of T^{st}_2.

Construction step 1:

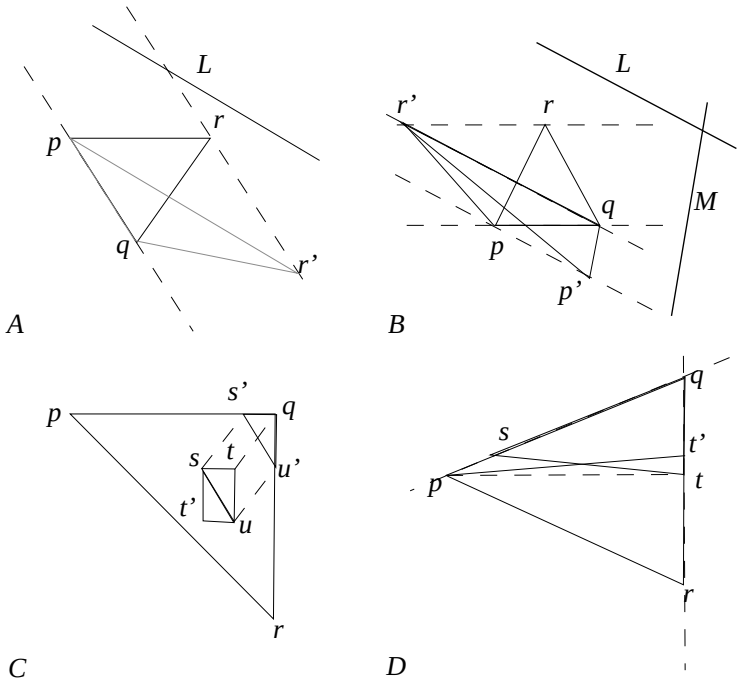


Figure 7.8: Area-preserving affine-invariant constructions.

Given a triangle T^{st}_{pqr} and a line L, we construct a triangle with the same area as T^{st}_{pqr}, but one side parallel to L. We do this by moving the point r over the line through r and parallel with pq until one of the line segments pr or qr is parallel to L. The resulting triangle $T^{st}_{pqr'}$ has the same area as T^{st}_{pqr} because it has the same base line segment and the same height as T^{st}_{pqr}. Figure 7.8, part A, illustrates this construction.

If we apply this construction twice, we can construct a triangle with two sides parallel to two given (different) lines. This is shown in Figure 7.8, part B, where the triangle T^{st}_{pqr} is first transformed into $T^{st}_{pqr'}$ and, in a second step, into $T^{st}_{p'qr'}$.

Construction Step 2:

Let a triangle T^{st}_{pqr} and a smaller triangle, either T^{st}_{stu} or $T^{st}_{st'u}$, which has two sides parallel to the sides pq and qr respectively of T^{st}_{pqr}, be given. There are two possible orientations for the smaller triangle. Either it is oriented in such a way that the corner point t' is on the opposite side of su than the point q, as is the case for triangle $T^{st}_{st'u}$ in Figure 7.8, part C, or it is oriented otherwise, as is the case for triangle T^{st}_{stu}. In the first case, we *flip* $T^{st}_{st'u}$ by constructing the parallelogram $st'ut$, and then considering the triangle T^{st}_{stu}.

Next, starting from a triangle T^{st}_{stu} with the right orientation, we construct a triangle $T^{st}_{qs'u'}$ which has the same area as T^{st}_{stu}, but shares a corner point with T^{st}_{pqr} and has its other corner points on the two sides of T^{st}_{pqr}, adjacent to the common corner point. This transformation involves only a translation, which can be carried out by constructing a set of parallel lines.

Construction Step 3:

Given a triangle T^{st}_{pqr}, and a triangle T^{st}_{sqt} such that s lies on the line through pq and t lies on the line through qr. We can construct a triangle $T^{st}_{pqt'}$ that has the same area as T^{st}_{sqt} by making sure that the cross-ratio of the points p, s and q equals the cross-ratio of the points t, t' and q. Figure 7.8, part D, illustrates this construction.

Using the above three steps, we constructed, starting from two arbitrary triangles, one being part of the other, two triangles that have the desired configuration.

We now can prove that our alternative language, FO($\{\mathbf{PartOf}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{Lex}\}$) also is sound and complete for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO-queries on triangle databases. As the proof is completely analog as the proof of Theorem 7.35, except for the translations of the predicates **Lex** and $\mathbf{EqCr}^{ST}$, we only give those translations.

Theorem 7.38 (**Expressiveness of** FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Lex}\}$)). Let $\hat{\sigma}^{st}$ be a spatio-temporal triangle database schema. The language FO($\{\mathbf{PartOf}^{\mathrm{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{Lex}\}$, $\hat{\sigma}^{st}$) is sound and complete for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO(+, ×, <, 0, 1)-queries on spatio-temporal triangle databases.

Proof. First, let $\hat{\sigma}^{st} = \{\hat{R}^{st}_1, \hat{R}^{st}_2, \ldots, \hat{R}^{st}_m\}$ be a spatio-temporal triangle database schema and let $\dot{\sigma}^{st} = \{\dot{R}^{st}_1, \dot{R}^{st}_2, \ldots, \dot{R}^{st}_m\}$ be a spatio-temporal point database schema where the arity of $\dot{R}^{st}_i$ is $3 \times ar(\hat{R}^{st}_i)$, for $i = 1, 2, \ldots, m$.

We first show that the predicate **Lex** can be expressed in FO($\{\mathbf{Between}^{\mathrm{Cotemp}}, \mathbf{Before}, \mathbf{EqCr}^{ST}\}$, $\dot{\sigma}^{st}$) . We verify that this predicate is invariant for transformations in $(\mathcal{AC}_{st}, \mathcal{A}_t)$. The proportion of the areas of two co-temporal triangles is invariant

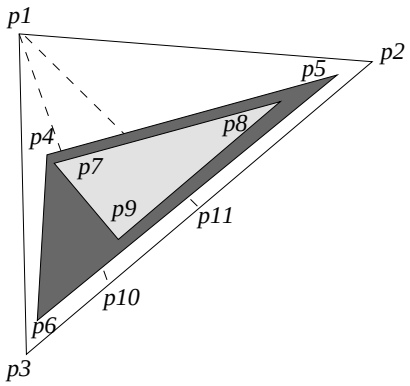


Figure 7.9: An illustration of the predicate **SameRelArea**. The expression **SameRelArea**$(p_1, p_2, \ldots, p_{11})$ will be true if and only if two conditions are met. First, the triangle with corner points p_7, p_8 and p_9 (the light shaded one) is part of the triangle with corner points p_4p, p_5 and p_6 (the dark shaded one), which is part of the triangle with corner points p_1, p_2 and p_3 (the white triangle). Second, The areas of the light and dark shaded triangles are to the area of the white triangle as the areas of the triangles with corner points p_1, p_2 and p_{10}, resp. p_1, p_2 and p_{11} to the area of the white triangle.

under affinities. This, together with the fact that cross-ratios of time moments are invariant under affine transformations of the time, shows that the predicate **Lex** is $(\mathcal{AC}_{st}, \mathcal{A}_t)$-invariant.

The constructions described in Observation 7.4.2 can all be expressed in the language FO($\{$**Between**$^{\text{Cotemp}}$, **Before**, **EqCr**$^{ST}\}$). They mainly involve parallelism-constraints on points.

Let **SameRelArea** be the abbreviation for a predicate in FO($\{$**Between**$^{\text{Cotemp}}$, **Before**, **EqCr**$^{ST}\}$) of arity 11. The first nine free variables represent the corner points of three co-temporal triangles, such that the first triangle is part of the second, which is again part of the third triangle. The two last point variables are located on one side of the third triangle, in such a way that the parts they define of the third triangle (denoted triangle four and five), are part of each other also. Finally, the proportion of the areas of the first three triangles is the same as the proportion of the areas of the fourth, fifth and third triangle. Fig 7.9 illustrates this predicate.

The translation of **Lex**$(\triangle_1^{st}, \triangle_2^{st}, \ldots, \triangle_6^{st})$ then is the following expression:

$$\exists v_1\, \exists v_2\, (\mathbf{SameArea}(u_{1,1}, u_{1,2}, u_{1,3}, u_{2,1}, u_{2,2}, u_{2,3}, u_{3,1}, u_{3,2}, u_{3,3}, v_1, v_2) \wedge \mathbf{EqCr}^{ST}(v_1, v_2, u_{3,3}, u_{4,1}, u_{5,1}, u_{5,2})),$$

if $\triangle_i^{st}$ is translated by $u_{i,1}$, $u_{i,2}$ and $u_{i,2}$ for $i = 1, \ldots, 6$.

The translation in the other direction is simpler. The formula **EqCr**$^{ST}(u_1, u_2, u_3,$

u_4, u_5, u_6) can be expressed as

$$\exists\, \triangle_7^{st}\ \exists\, \triangle_8^{st}\ \exists\, \triangle_9^{st}\ \exists\, \triangle_{10}^{st}\ \exists\, \triangle_{11}^{st}\ (\\ \mathbf{CornerP}(\triangle_7^{st}, \triangle_8^{st}, \triangle_1^{st}, \triangle_9^{st}) \wedge \mathbf{CornerP}(\triangle_7^{st}, \triangle_8^{st}, \triangle_2^{st}, \triangle_{10}^{st}) \\ \wedge\, \mathbf{CornerP}(\triangle_7^{st}, \triangle_8^{st}, \triangle_3^{st}, \triangle_{11}^{st}) \wedge \mathbf{Lex}(\triangle_9^{st}, \triangle_{10}^{st}, \triangle_{11}^{st}, \triangle_4^{st}, \triangle_5^{st}, \triangle_6^{st})).$$

□

7.4.2 Physics-based Classes

In the previous section, we investigated a triangle language for $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic triangle queries. Next, we focus on triangle languages for the physics-based queries, *i.e.*, those generic for the group $(\mathcal{V}_{st}, \mathcal{A}_t)$ of velocity-preserving transformations and the group $(\mathcal{AC}_{st}, \mathcal{A}_t)$ of acceleration-preserving transformations.

In Chapter 4, the query languages expressing queries generic for the physics-based transformation groups were found by starting with the languages expressing the affine-invariant spatial point queries. The reason was that the physics-based transformation groups of $((\mathbb{R}^2 \times \mathbb{R}))$ are a subgroup of the affinities of $\mathbb{R}^3$, and that spatio-temporal points in $((\mathbb{R}^2 \times \mathbb{R}))$ can be interpreted equally well as points in $\mathbb{R}^3$.

Here, it is not expedient to do so. We can see spatio-temporal triangles in $((\mathbb{R}^2 \times \mathbb{R}))$ as convex objects in $\mathbb{R}^3$, but then the predicate **PartOf** would not make much sense, as spatio-temporal triangles can only overlap when they exist at the same moment in time. Another solution would be to choose other convex objects, that have a temporal extend of more than one time moment. But, these objects would make rather poor spatio-temporal objects. Indeed, even if all corner points of a triangle in $\mathbb{R}^2$ move with a linear function of time, this movement can result in a 3-dimensional object bounded by non-planar surfaces, and hence possibly not convex.

Therefor, we take another approach and start with the predicates $\mathbf{PartOf}^{\text{Cotemp}}$ and $\mathbf{Before}_{\triangle}$, as in the previous section, and add other predicates until the resulting language is expressive enough. In concrete, this means that we have to be able to translate the point predicate $\mathbf{Between}^{(n+1)}$ in that language.

As $(\mathcal{V}_{st}, \mathcal{A}_t) \subset (\mathcal{AC}_{st}, \mathcal{A}_t)$, we start with the acceleration preserving transformations first, and later extend the language expressing all $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic queries in such a way we obtain a language expressing the $(\mathcal{V}_{st}, \mathcal{A}_t)$-generic queries.

7.4.2.1 $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic Queries

For the acceleration-preserving queries, we introduce the spatio-temporal triangle predicate **SAS** (which is an abbreviation for "Same Average Speed"). Let $\triangle_1^{st}, \triangle_2^{st}$ $, \triangle_3^{st}$ and $\triangle_4^{st}$ be four triangles that have center of mass $p_i = (a_i, b_i, \tau_i), i = 1 \ldots 4$. Furthermore, $\tau_1 \leq \tau_2$ and $\tau_3 \leq \tau_4$. Then $\mathbf{SAS}(\triangle_1^{st}, \triangle_2^{st}, \triangle_3^{st}, \triangle_4^{st})$ is true if and only if

$$\frac{a_2 - a_1}{\tau_2 - \tau_1} = \frac{a_4 - a_3}{\tau_4 - \tau_3} \text{ and } \frac{b_2 - b_1}{\tau_2 - \tau_1} = \frac{b_4 - b_3}{\tau_4 - \tau_3}.$$

In other words, the movement from $\triangle_1^{st}$ to $\triangle_2^{st}$ has the same average speed, in both x- and y-direction, as the movement from $\triangle_3^{st}$ to $\triangle_4^{st}$.

We now show that the language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$) is sound and complete for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO(+, ×, <, 0, 1)-queries on triangle databases.

As soundness and completeness proof are completely analogous to those of the previous section, we only give the translations of the triangle predicates from the set $\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$ into FO($\{\mathbf{Between}^{(2+1)}\}$) and of $\mathbf{Between}^{(2+1)}$ into FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$).

Theorem 7.39 (**Expressiveness of the language** FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$) **).** Let $\hat{\sigma}^{st}$ ba a triangle database schema. Let $\overline{\sigma}^{st}$ be the corresponding semi-algebraic database schema. The language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$, $\hat{\sigma}^{st}$) is sound and complete for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO(+, ×, <, 0, 1, $\overline{\sigma}^{st}$)-queries on triangle databases over $\hat{\sigma}^{st}$.

Proof sketch.
Let $\hat{\sigma}^{st} = \{\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}\}$ be a spatial triangle database schema. Let $\dot{R}_i^{st}, 1 \leq i \leq m$ be the corresponding spatial point relation names of arity $3 \times ar(\hat{R}_i^{st})$ and let $\dot{\sigma}^{st}$ be the spatial database schema $\{\dot{R}_1^{st}, \dot{R}_2^{st}, \ldots, \dot{R}_m^{st}\}$. Let $\overline{R}_i^{st}, 1 \leq i \leq m$ be the corresponding constraint relation names of arity $6 \times ar(\hat{R}_i^{st})$ and let $\overline{\sigma}^{st}$ be the spatial database schema $\{\overline{R}_1^{st}, \overline{R}_2^{st}, \ldots, \overline{R}_m^{st}\}$.

In this proof sketch, we only give the translation of **SAS** into FO($\{\mathbf{Between}^{(2+1)}\}$, $\dot{\sigma}^{st}$). For the translations of $\mathbf{PartOf}^{\text{Cotemp}}$ and $\mathbf{Before}_{\triangle}$, see Section 7.3 and Section 7.4.1 respectively.

Given the expression $\mathbf{SAS}(\triangle_1^{st}, \triangle_2^{st}, \triangle_3^{st}, \triangle_4^{st})$. The following formula is its translation into FO($\{\mathbf{Between}^{(2+1)}\}$, $\dot{\sigma}^{st}$):

$$\exists v_1\, \exists v_2\, \exists v_3\, \exists v_4\, (\bigwedge_{i=1}^{4} \mathbf{CenterOM}(v_i, u_{i,1}, u_{i,2}, u_{i,3}) \wedge \mathbf{Before}(v_1, v_2) \wedge \mathbf{Before}(v_3, v_4) \wedge \mathbf{CoPlanar}(v_1, v_2, v_3, v_4) \wedge \neg \exists w\, (\mathbf{Collinear}(w, v_1, v_2) \wedge \mathbf{Collinear}(w, v_3, v_4))).$$

We have omitted the sub formulas expressing that the corner points of a triangle should be co-temporal. The predicate **CoPlanar** expresses that four 3-dimensional points are co-planar. It is clear that this is an affine invariant and FO-expressible.

For the definition of **CenterOM**, we refer to the proof of Lemma 7.36.

We next prove that the predicate $\mathbf{Between}^{(2+1)}$ can be expressed in the language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$, $\hat{\sigma}^{st}$). This translation is not complicated. If the expression $\mathbf{Between}^{(2+1)}(p, q, r)$ holds for three points p, q and r, then either they are all co-temporal or they all exist at a different time moment. In the first case, we can translate **Between** using **PartOf**, as we showed in the proof of Lemma 7.17. If they all have a different time coordinate, we can express that q is between p and r using **SAS**:

$$(\mathbf{CoTemp}(\triangle_1^{st}, \triangle_2^{st}) \wedge \mathbf{CoTemp}(\triangle_2^{st}, \triangle_3^{st}) \wedge \mathbf{Between}_{\Delta}(\triangle_p, \triangle_q, \triangle_r)) \vee (\mathbf{SAS}(\triangle_1^{st}, \triangle_2^{st}, \triangle_2^{st}, \triangle_3^{st})).$$

In the previous formula, we have omitted the sub-formulas expressing that the triangles translating the point variables should be points. □

Since the group $(\mathcal{V}_{st}, \mathcal{A}_t)$ is a subgroup of the group $(\mathcal{AC}_{st}, \mathcal{A}_t)$, we use our knowledge from this subsection to extend the language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$, $\hat{\sigma}$), which we will do next.

7.4.2.2 $(\mathcal{V}_{st}, \mathcal{A}_t)$-generic Queries

In this subsection, we propose a language sound and complete of the first-order $(\mathcal{V}_{st}, \mathcal{A}_t)$-generic triangle queries. We add the element **NoSp** (which is an abbreviation for "No Speed") to the predicate set $\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}\}$.

Suppose two spatio-temporal triangles T_1^{st} and T_2^{st} have center of mass $p_i = (a_i, b_i, \tau_i), i = 1, 2$. If we furthermore assume that $\tau_1 \leq \tau_2$, then $\mathbf{NoSp}(\triangle_1^{st}, \triangle_2^{st})$ is true if and only if $a_1 = a_2$ and $b_1 = b_2$. In other words, the average speed is zero, both triangles are on the same position.

We now show that the language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}, \mathbf{NoSp}\}$) is sound and complete for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO(+, ×, <, 0, 1)-queries on triangle databases.

As soundness and completeness proof are completely analogous to those of the previous section, we only give the new translations.

Theorem 7.40 (**Expressiveness of the language** FO($\{\mathbf{PartOf}^{\mathbf{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}, \mathbf{NoSp}\}$) **).** Let $\hat{\sigma}^{st}$ be a triangle database schema. Let $\overline{\sigma}^{st}$ be the corresponding semi-algebraic database schema. The language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}, \mathbf{NoSp}\}$, $\hat{\sigma}^{st}$) is sound and complete for the $(\mathcal{AC}_{st}, \mathcal{A}_t)$-generic FO(+, ×, <, 0, 1, $\overline{\sigma}^{st}$)-queries on triangle databases over $\hat{\sigma}^{st}$.

Proof sketch.
Let $\hat{\sigma}^{st} = \{\hat{R}_1^{st}, \hat{R}_2^{st}, \ldots, \hat{R}_m^{st}\}$ be a spatial triangle database schema. Let $\dot{R}_i^{st}, 1 \leq i \leq m$ be the corresponding spatial point relation names of arity $3 \times ar(\hat{R}_i^{st})$ and let $\dot{\sigma}^{st}$ be the spatial database schema $\{\dot{R}_1^{st}, \dot{R}_2^{st}, \ldots, \dot{R}_m^{st}\}$. Let $\overline{R}_i^{st}, 1 \leq i \leq m$ be the corresponding constraint relation names of arity $6 \times ar(\hat{R}_i^{st})$ and let $\overline{\sigma}^{st}$ be the spatial database schema $\{\overline{R}_1^{st}, \overline{R}_2^{st}, \ldots, \overline{R}_m^{st}\}$.

In this proof sketch, we only give the translation of the predicate **NoSp** into FO($\{\mathbf{Between}^{(2+1)}, \mathbf{Before}, \mathbf{EqSpace}\}$) and of the predicate **EqSpace** into the language FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}, \mathbf{NoSp}\}$).

The next formula, with free variables $u_1, u_2, u_3, v_1, v_2, v_3$ is the translation of $\mathbf{NoSp}(\triangle_u, \triangle_v)$ into FO($\{\mathbf{Between}^{(2+1)}, \mathbf{Before}, \mathbf{EqSpace}\}$, $\hat{\sigma}$).

$$\exists w_1 \exists w_2 (\mathbf{CenterOM}(w_1, u_1, u_2, u_3) \wedge \mathbf{CenterOM}(w_2, v_1, v_2, v_3) \wedge \mathbf{EqSpace}(w_1, w_2)).$$

Finally, the formula

$$\mathbf{Point}(\triangle_u^{st}) \wedge \mathbf{Point}(\triangle_v^{st}) \wedge \mathbf{NoSp}(\triangle_u^{st}, \triangle_v^{st})$$

translates $\mathbf{EqSpace}(u, v)$ into FO($\{\mathbf{PartOf}^{\text{Cotemp}}, \mathbf{Before}_{\triangle}, \mathbf{SAS}, \mathbf{NoSp}\}$). Note that, if a triangle is degenerated into a point, its center of mass is equal to the triangle itself. □

8

Conclusion

In this thesis, we started with defining spatio-temporal databases as a special type of constraint databases, meaning that we defined spatio-temporal objects to be semi-algebraic subsets of $(\mathbb{R}^n \times \mathbb{R})$. A consequence of this choice is that spatio-temporal data can be described by Boolean combinations of polynomial equations, which can be expressed in the language FO(+, ×, <, 0, 1).

Although spatio-temporal databases have been studied for a decade already, the question "What are spatio-temporal database queries" is still actual. In relational database theory, it is common to require queries to be *generic*. A generic query asks only for properties that are shared by "isomorphic" encodings of the same data or, in other words, the result of a generic query depends only to a certain, limited extent on the actual internal representation of the database it is applied to. Chandra and Harel [11] considered the permutations of the universal domain $\mathcal{U}$ of the database (that possibly fix some elements of the domain) as "isomorphisms" for relational databases. Paredaens, Van den Bussche and Van Gucht [58] have shown that for spatial databases, the definition of genericity is not unique and that it depends on the particular kind of geometry in which the spatial information is to be interpreted. Genericity of spatial databases hence is defined as a function of some group of geometric transformations.

Spatio-temporal Genericity

We investigated which notions of genericity are appropriate for spatio-temporal databases and which transformation groups express them. In Chapter 4, we proposed a hierarchy of genericity classes for spatio-temporal databases. We observed that the transformations should first and foremost respect the monotone and unidirectional nature of time, *i.e.*, leave the temporal order of events unchanged. It followed that the relevant transformation groups are the product of a group of time-(in)dependent spatial transformations and a group of monotone increasing transformations of the

time-component of the spatio-temporal data.

First, we showed that all the genericity classes are undecidable, but that the considered classes of generic first-order queries are recursively enumerable, however. Hereto, we defined first-order point-based languages in which variables are assumed to range over points in $(\mathbb{R}^n \times \mathbb{R})$ and which contain certain point predicates (such as $\mathbf{Between}^{(n+1)}$ and **Before**). We showed that these point-based languages are sound and complete for the first-order queries in the considered genericity classes. We also showed that extensions of these point-based logics with a While-loop give sound and complete languages for the computable queries in the different genericity classes. Some results were obtained by techniques introduced by Gyssens, Van den Bussche and Van Gucht [43], but for time-dependent transformations we have introduced new proof techniques.

For what concerns computationally complete languages these techniques seem to be insufficient to deal with the genericity notions that are expressed by the groups $(\mathcal{A}^f_{st}, \mathcal{A}_t)$, $(\mathcal{A}^f_{st}, \mathcal{I}_t)$, $(\mathcal{A}^f_{st}, Id_t)$, $(\mathcal{S}^f_{st}, \mathcal{F}_t)$, $(\mathcal{I}^f_{st}, \mathcal{F}_t)$, and $(\mathcal{T}^f_{st}, \mathcal{F}_t)$, for $\mathcal{F}_t \in \{\mathcal{A}_t, \mathcal{T}_t, Id_t\}$. The problem in adapting the proof technique of Theorem 4.39 to these groups is that it is not clear how we can express in the respective point-based logics that two spatio-temporal databases can be mapped to each other by some piece-wise constant affinity. Indeed, since the number of pieces is not defined *a priori*, this might not be expressible. This would imply that yet another new proof technique would be required to deal with the remaining cases.

A limitation of the constraint model that possibly is more apparent in the context of spatio-temporal databases than it was for spatial databases, is the fact that the movement of spatio-temporal objects is restricted to rational functions of time. A helicopter flying at constant speed in circles around some event, cannot be modelled. This would require the use of the sin and cos functions. It would be possible however to model a helicopter flying in circles, albeit not at constant speed. This is possible because *conic sections* can be parameterized by rational functions of t [32].

Parametric Spatio-temporal Objects

The constraint model allowed us to describe a wide range of spatio-temporal phenomena, but, users tend to think about spatio-temporal data as "moving objects" rather than sets in $(\mathbb{R}^n \times \mathbb{R})$. Therefor, we investigated a more concrete data model in Chapter 5. We introduced the concept of spatio-temporal *object* to model events and objects that change in time. More general objects can be constructed from basic, or *atomic* objects using the union operation. For a variety of special classes of *spatio-temporal objects* of practical relevance, we investigated their closure properties with respect to Boolean set operators. An exhaustive study of these closure properties shows that the chosen approach leads to mostly negative closure results, except for the class of scaling rectangles (that is developed further in [62]) and the class of triangles that are moving by rational affinities. Therefore, we proposed an adaptation to the model, which led to better closure properties.

To implement our approach, it is sufficient to be able to represent in a database the following:

- spatial objects (a solved problem for many classes of such objects),

- temporal objects (again a solved problem),
- function objects (rational functions can be represented as lists of coefficients. For linear polynomials, such lists are of fixed length, opening the possibility of representing the corresponding spatiotemporal objects using the standard relational data model).

In addition to implementation issues, it would be challenging to develop a *type system* that captures different *dimensions* of specialization present in geometric objects: region specialization (polygon, rectangle, ...), transformation specialization (affine mapping, scaling, ...) and time function specialization (rational, polynomial, ...).

Spatial and Spatio-temporal Triangulations

We discussed the need for a normal form in the context of the adapted data model, where also intersection and difference were allowed, in addition to union, to construct more complex objects out of atomic objects. But, also the original data model, where only union is used to construct more complex objects, would benefit from a normal form. Indeed, a drawback of modelling an object as an arbitrary set of atomic objects, is that it is not clear immediately how the spatio-temporal object represented by the atomic objects looks like. Its time domain has to be computed from the time domains of all atomic objects, which might overlap. Also, there may be gaps, *i.e.*, moments when the spatio-temporal object does not exist, and two sets of atomic objects can represent the same spatio-temporal object. Or, there might be elements in the set of atomic objects that do not contribute to the spatio-temporal object at all, as they are overlapped totally by other atomic objects.

We developed a normal form for the class of triangles that are moving by rational affinities. Because the spatial reference objects are triangles, a spatio-temporal triangulation would seem a natural normalization. Motivated by the importance of affine-invariance in the computer graphics, robotics and computer vision communities (expressed in the *weak perspective assumption*), we developed an affine-invariant spatio-temporal triangulation. To our knowledge, only one affine-invariant triangulation for spatial data exists yet [56].

Therefor, we first developed an affine-invariant spatial triangulation method. We then "extended" this affine-invariant spatial triangulation to an time-dependent affine-invariant spatio-temporal triangulation. This spatio-temporal triangulation is a special type of spatio-temporal partition. A formal definition of spatio-temporal partitions was proposed by Erwig and Schneider [29]. However, they did not give a concrete example of such a spatio-temporal partition or partitioning method. The proposed method is, to our knowledge, the first spatio-temporal partitioning method.

An interesting follow-up to this spatio-temporal triangulation would be to develop an affine-invariant way of *storing the result* of this triangulation. Knowledge of this affine-invariant representation only would naturally limit users to posing affine-generic queries.

We now describe some applications that we believe can benefit from the spatio-temporal triangulation.

- *Efficient rendering of objects:* When a geometric object that is not in normal form has to be displayed to the user, the displaying algorithm would have to

keep track of the time domains of the individual atomic objects and keep a list of *active* ones at the moment under consideration, which has to be updated every instant. If the geometric object is in normal form, the atomic objects can be sorted by their time domains, and during each interval in the partition of time domain, the list of active atomic objects will remain the same.

Also, if the geometric object is not in normal form, the snapshots of the atomic objects may overlap, so pixels will be computed more than once. When a geometric object is in normal form, no triangles overlap, so each pixel will be computed only once.

- *Moving object retrieval:* The triangulation provides a means of automatic affine invariant feature extraction for moving object recognition. Indeed, the number of intervals in the time domain indicates the complexity of the movement of the geometric object. This can be used as a first criterium for object matching. For objects having approximately the same number of intervals in their time domains, the snapshots at the middle of each time interval can be compared. If they are all similar, which can be, for example, defined as $\mathcal{T}_S$-isomorphic, the objects match.

- *Surveillance Systems:* In some applications, e.g., surveillance systems, it is important to know the time moments when something changed, when some discontinuity appeared. This could mean that an unauthorized person entered a restricted area, for example, or that a river has burst its banks. Triangulating the contours of the recorded images and reporting all single points and end points of intervals of the partition of the time domain indicates all moments when some discontinuity might have occurred.

- *Pre-computing queries:* The atomic objects in the triangulation of a geometric object which is, essentially the union of a set of geometric objects having different labels will have the following nice property. We can label each atomic element of the spatio-temporal triangulation of the database with the set of id's of the geometric objects it belongs to. We illustrate this for the spatial case only in Figure 8.1. Suppose we have two triangles A and B. The set A is the union of the light grey and white parts of the figure, the set B is the union of the dark grey and white parts of the figure. After triangulation, we can label the light grey triangles with $\{A\}$, the white triangle with $\{A, B\}$, and the dark grey triangles with $\{B\}$.

  Using this triangulation of databases in a preprocessing stage, means that the results of queries that ask for set operations between geometric objects are also pre-computed. A lot of spatio-temporal queries essentially involve set operations.

We end the paragraph on triangulations with a note on maintaining the triangulation. If a geometric object has to be inserted into or removed from a database (*i.e.*, a collection of geometric objects), the triangulation has to be recomputed for the intervals in the partition of the time domain that contain the time domain of the

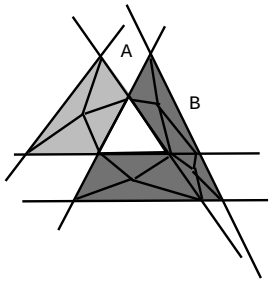


Figure 8.1: The set operations between objects are pre-computed in the triangulation.

object under consideration. This may require that the triangulation in total has to be recomputed.

However, the nature of a lot of spatio-temporal applications is such that updates involve only the insertion of objects that exists *later* in time than the already present data. In that case, only the triangulation at the latest time interval of the partition should be recomputed together with the new object, to check whether the new data are a continuation of the previous. Also, data is removed only when it is outdated. In that case a whole time interval of data can be removed. Examples of such spatio-temporal applications are surveillance, traffic monitoring and cadastral information systems.

Triangle-based Languages

Although the results of Chapter 4 are important to understand the nature of spatio-temporal database queries, it turned out that the generic spatio-temporal query languages we proposed were not very intuitive, and hence not of great practical use. We simplified the data model from semi-algebraic sets in $(\mathbb{R}^n \times \mathbb{R})$ to sets of triangles, that can be described using the constraint model. We investigated whether knowledge about the nature of the data, *i.e.*, collections of triangles, could lead to more intuitive languages.

When we were developing spatio-temporal point languages in Chapter 4, we could build on the previous work of Gyssens, Van den Bussche and Van Gucht [43] on spatial databases. Spatial triangle-based languages were not previously studied, so we started there. Given the fact that a spatial database contains a (possible infinite) set of triangles, we obtained a triangle-based affine-invariant query language only containing the binary predicate **PartOf**. Safety of queries in this language is not guaranteed, *i.e.*, the result of a query on a finite collection of triangles is possibly an infinite collection of triangles. We showed that it is decidable whether the result of a query is finitely representable, however, and that this can be decided by a formula in the triangle language.

At this point, we want to mention an interesting direction for further work on triangle-based languages. One of the main reasons that the triangle query language we propose is not safe, is that one can, using an infinite collection of triangles, represent a non-linear figure. Vandeurzen et al. [73] developed languages capturing exact the linear queries definable with polynomial constraints. A triangle-based query language

that has the same expressiveness as the affine-invariant fragment of the language of Vandeurzen et al, would be safe.

We also found that the predicate **PartOf** can be generalized to $\mathbf{PartOf}^{(n,k)}$, which arguments are n-dimensional convex objects with k corner points, abbreviated as (n,k)-objects, and that the language $\mathrm{FO}(\{\mathbf{PartOf}^{(n,k)}\})$ is sound and complete for the first-order affine-generic queries on (n,k)-objects. This finding is particularly interesting in comparison with the work of Aiello and van Benthem [3, 4] on modal logics of space. They proposed a topological modal logic over regions, which can express "connectedness" and "parthood". By adding a "convexity" operator (expressed using a "betweenness" operator), they obtained an affine modal logic. Essentially, we obtain the same result for first-order logic, as triangles (or (n,k)-objects) are convex and connected regions, and we add the "parthood" operator **PartOf**.

After investigating spatial triangle-based queries, we continued with the actual purpose, developing spatio-temporal triangle-based languages, generic for the time-dependent affinities, the velocity-preserving and acceleration preserving transformations, respectively. For the time-dependent affinities, we developed two equally expressive languages. The first one can be used when data is interpreted as objects that are retaining their shape and move along a curve. The second language is more useful when data changes its shape. For the acceleration and velocity-preserving transformations, we proposed triangle predicates expressing that objects have linear speed or no speed at all.

Bibliography

[1] D. Abel and B. C. Ooi, editors. *Proceedings of the 3rd International Symposium on Spatial Databases*, volume 692 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[2] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[3] M. Aiello and J. van Benthem. Logical patterns in space. In D. Barker-Plummer, D. Beaver, J. van Benthem, and P. Scotto di Luzio, editors, *Words, Proofs, and Diagrams*, pages 5–25. CSLI, 2002.

[4] M. Aiello and J. van Benthem. A modal walk through space. *Journal of Applied Non-Classical Logics*, 12(3-4):319–363, 2002.

[5] J. F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[6] J. F. Allen and G. Ferguson. Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

[7] M. Benedikt and L. Libkin. Safe constraint queries. *SIAM Journal on Computing*, 29(5):1652–1682, 2000.

[8] J. Bochnak, M. Coste, and M.F. Roy. *Géométrie Algébrique Réelle*. Springer-Verlag, Berlin, 1987.

[9] A. Buchmann, editor. *Proceedings of the 1st International Symposium on Spatial Databases*, volume 409 of *Lecture Notes in Computer Science*. Springer-Verlag, 1989.

[10] M. Cai, D. Keshwani, and P. Revesz. Parametric rectangles: A model for querying and animation of spatiotemporal databases. In C. Zaniolo, P. C. Lockemann, M. H. Scholl, and T. Grust, editors, *EDBT*, volume 1777 of *Lecture Notes in Computer Science*, pages 430–444. Springer, 2000.

[11] A. K. Chandra and D. Harel. Computable queries for relational data bases. *Journal of Computer and System Sciences*, 21(2):156–178, 1980.

[12] C. X. Chen and C. Zaniolo. Universal temporal extensions for database languages. In *Proceedings of the 15th International Conference on Data Engineering*, pages 428–437. IEEE Computer Society, 1999.

[13] C. X. Chen and C. Zaniolo. SQLST: A spatio-temporal data model and query language. In V. C. Storey A. H. F. Laender, S. W. Liddle, editor, *19th International Conference on Conceptual Modeling*, volume 1920 of *Lecture Notes in Computer Science*, pages 96–111. Springer-Verlag, 2000.

[14] J. Chomicki and P. Revesz. A geometric framework for specifying spatiotemporal objects. In *Proceedings of the 6th International Workshop on Temporal Representation and Reasoning*, pages 41–46. IEEE Computer Society, 1999.

[15] M. Coste. *An Introduction to Semialgebraic Geometry*. PhD thesis, Dip. Mat. Univ. Pisa, Istituti Editoriali e Poligrafici Internazionali, Pisa, 2000.

[16] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Arrangements and duality. In *Computational Geometry: Algorithms and Applications*, chapter 8, pages 165–182. Springer-Verlag, 2000.

[17] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Computational geometry. In *Computational Geometry: Algorithms and Applications*, chapter 1, pages 1–17. Springer-Verlag, 2000.

[18] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 2000.

[19] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. Line segment intersection. In *Computational Geometry: Algorithms and Applications*, chapter 2, pages 18–43. Springer-Verlag, 2000.

[20] E. Desloge. *Classical Mechanics*. Wiley, New York, 1982.

[21] F. Dumortier, M. Gyssens, L. Vandeurzen, and D. Van Gucht. On the decidability of semi-linearity for semi-algebraic sets and its implications for spatial databases. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, pages 68–77. ACM Press, 1997.

[22] H. Edelsbrunner, L. J. Guibas, and J. Stolfi. Optimal point location in a monotone subdivision. *SIAM J. Comput.*, 15(2):317–340, 1986.

[23] M. Egenhofer and R. Golledge. Time in geographic space, report on the specialist meeting of research initiative 10. Technical Report 94-9, National Center for Geographic Information and Analysis, Univeristy of California, Santa Barbara, 1994.

[24] M. Egenhofer and J. Herring. A mathematical framework for the definition of topological relationships. In K. Brassel and H. Kishimoto, editors, *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 803–813, 1990.

[25] M. J. Egenhofer and R. D. Franzosa. Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174, 1991.

[26] M. J. Egenhofer and J. R. Herring, editors. *Proceedings of the 4th International Symposium on Spatial Databases*, volume 951 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[27] 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*, 3(3):269–296, 1999.

[28] M. Erwig and M. Schneider. Partition and conquer. In *Proceedings of the 3rd International Conference on Spatial Information Theory*, volume 1329 of *Lecture Notes in Computer Science*, pages 389–408. Springer, 1997.

[29] M. Erwig and M. Schneider. The honeycomb model of spatio-temporal partitions. In M. H. Böhlen, C. S. Jensen, and M. Scholl, editors, *Proceedings of the International Workshop on Spatio-Temporal Database Management*, volume 1678 of *Lecture Notes in Computer Science*, pages 39–59. Springer, 1999.

[30] M. Erwig and M. Schneider. Spatio-temporal predicates. *IEEE Trans. Knowl. Data Eng.*, 14(4):881–901, 2002.

[31] A. Frank, S. Grumbach, R. Güting, C. Jensen, M. Koubarakis, N. Lorentzos, Y. Manopoulos, E. Nardelli, B. Pernici, H.-J. Schek, M. Scholl, T. Sellis, B. Theodoulidis, and P. Widmayer. Chorochronos: A research network for spatiotemporal database systems. *SIGMOD Record*, 28:12–21, 1999.

[32] W. Fulton. *Algebraic curves: an introduction to algebraic geometry*. W. A. Benjamin, Inc., New York, 1969.

[33] J. Goldfeather, S. Molnar, G. Turk, and H. Fuchs. Near real-time csg rendering using tree normalization and geometric pruning. *IEEE Computer Graphics and Applications*, 9(3):20–28, 1989.

[34] T. Griffiths, A. A. A. Fernandes, N. W. Paton, and R. Barr. The tripod spatio-historical data model. *Data Knowl. Eng.*, 49(1):23–65, 2004.

[35] S. Grumbach, P. Rigaux, M. Scholl, and L. Segoufin. The dedale prototype. In *Constraint Databases*, pages 365–382, 2000.

[36] S. Grumbach, P. Rigaux, and L. Segoufin. On the orthographic dimension of constraint databases. In C. Beeri and P. Buneman, editors, *Proceedings of the 7th International Conference on Database Theory*, volume 1540 of *Lecture Notes in Computer Science*, pages 199–216. Springer, 1999.

[37] S. Grumbach and J. Su. Queries with arithmetical constraints. *Theor. Comput. Sci.*, 173(1):151–181, 1997.

[38] O. Gunther and H.-J. Schek, editors. *Proceedings of the 2nd International Symposium on Spatial Databases*, volume 525 of *Lecture Notes in Computer Science*. Springer-Verlag, 1991.

[39] R. H. Güting, M. H. Bohlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Databases Systems*, 25:1–42, 2000.

[40] Ralf Hartmut Güting and Markus Schneider. Realm-based spatial data types: The rose algebra. *VLDB Journal*, 4(2):243–286, 1995.

[41] R.H. Güting, editor. *Proceedings of the 6th International Symposium on Spatial Databases*, volume 1651 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[42] M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometrical query languages. In *Proceedings of the 16th ACM Symposium on Principles of Database Systems*, pages 62–67. ACM Press, 1997.

[43] M. Gyssens, J. Van den Bussche, and D. Van Gucht. Complete geometric query languages. *Journal of Computer and System Sciences*, 58(3):483–511, 1999.

[44] S. Haesevoets and B. Kuijpers. Time-dependent affine triangulation of spatio-temporal data. In D. Pfoser, I. F. Cruz, and M. Ronthaler, editors, *Proceedings of the 12th ACM International Workshop on Geographic Information Systems*, pages 57–66. ACM, 2004.

[45] C. M. Hoffmann. *Geometric and Solid Modeling: An Introduction*. Morgan Kaufmann, 1989.

[46] P. C. Kanellakis, G. M. Kuper, and P.Z. Revesz. Constraint query languages. In *Proceedings of the 9th ACM Symposium on Principles of Database Systems*, pages 299–313. ACM Press, 1990.

[47] P. C. Kanellakis, G. M. Kuper, and P.Z. Revesz. Constraint query languages. *Journal of Computer and System Sciences*, 51:26–52, 1995.

[48] M. Koubarakis, T. K. Sellis, A. U. Frank, S. Grumbach, R. H. Güting, C. S. Jensen, N. A. Lorentzos, Y. Manolopoulos, E. Nardelli, B. Pernici, H.-J. Schek, M. Scholl, B. Theodoulidis, and N. Tryfona, editors. *Spatio-Temporal Databases: The CHOROCHRONOS Approach*, volume 2520 of *Lecture Notes in Computer Science*. Springer, 2003.

[49] B. Kuijpers. Personal communication, 2003.

[50] B. Kuijpers and D. Van Gucht. Genericity in spatial databases. In J. Paredaens, G. Kuper, and L. Libkin, editors, *Constraint databases*, chapter 12, pages 293–304. Springer-Verlag, 2000.

[51] B. Kuijpers, J. Paredaens, and D. Van Gucht. Towards a theory of movie database queries. In *Proceedings of the 7th International Workshop on Temporal Representation and Reasoning*, pages 95–102. IEEE Computer Society, 2000.

[52] B. Kuijpers and M. Smits. On expressing topological connectivity in spatial datalog. In V. Gaede, A. Brodsky, O. Gunter, D. Srivastava, V. Vianu, and M. Wallace, editors, *Proceedings of the Second International Workshop on Constraint Database Systems*, volume 1191 of *Lecture Notes in Computer Science*, pages 116–133, Berlin, 1997. Springer-Verlag.

[53] L. Libkin. Some remarks on variable independence, closure, and orthographic dimension in constraint databases. *SIGMOD Record*, 28(4):24–28, 1999.

[54] Yu. V. Matijasevich. *Hilbert's Tenth Problem*. MIT Press, Cambridge, MA, 1993.

[55] H. Mokhtar, J. Su, and O. H. Ibarra. On moving object queries. In L. Popa, editor, *Proceedings of the 21st ACM Symposium on Principles of Database Systems*, pages 188–198. ACM, 2002.

[56] G. Nielson. A characterization of an affine invariant triangulation. In G. Farin, H. Hagen, and H. Noltemeier, editors, *Geometric Modelling, Computing Supplementum 8*, pages 191–210, 1993.

[57] E. Novak and K. Ritter. Some complexity results for zero finding for univariate functions. *Journal of Complexity*, 9:15–40, 1993.

[58] J. Paredaens, J. Van den Bussche, and D. Van Gucht. Towards a theory of spatial database queries. In *Proceedings of the 13th ACM Symposium on Principles of Database Systems*, pages 279–288, New York, 1994. ACM Press.

[59] J. Paredaens, G. Kuper, and L. Libkin, editors. *Constraint databases*. Springer-Verlag, 2000.

[60] C. Parent, A. Spaccapietra, and E. Zimányi. The murmur project: Modeling and querying multi-representation spatio-temporal databases. *Information Systems*, To appear, 2005.

[61] C. Parent, S. Spaccapietra, and E. Zimányi. Spatio-temporal conceptual models: Data structures + space + time. In C. B. Medeiros, editor, *Proceedings of the 7th ACM International Workshop on Geographic Information Systems*, pages 26–33. ACM, 1999.

[62] P. Revesz. *Introduction to Constraint Databases*. Springer-Verlag, 2002.

[63] P. Revesz and M. Cai. Efficient querying and animation of periodic spatio-temporal databases. *Ann. Math. Artif. Intell.*, 36(4):437–457, 2002.

[64] Ph. Rigaux, M. Scholl, and A. Voisard. *Introduction to Spatial Databases: Applications to GIS*. Morgan Kaufmann, 2000.

[65] L.G. Roberts. Machine perception of three-dimensional solids. In J.T. Tippett, editor, *Optical and Electro-Optical Information Processing*, chapter 9, pages 159–197. MIT Press, Cambridge, Massachusetts, 1965.

[66] M. Scholl and A. Voisard, editors. *Proceedings of the 5th International Symposium on Spatial Databases*, volume 1262 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[67] W. Schwabhäuser, W. Szmielew, and A. Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, 1983.

[68] A. P. Sistla, O. Wolfson, S. Chamberlain, and S. Dao. Modeling and querying moving objects. In *Proceedings of the 13th International Conference on Data Engineering*, pages 422–432. IEEE Computer Society, 1997.

[69] J. Su, H. Xu, and O. H. Ibarra. Moving objects: Logical relationships and queries. In C. S. Jensen, M. Schneider, B. Seeger, and V. J. Tsotras, editors, *Proceedings of the 7th International Symposium on Advances in Spatial and Temporal Databases*, volume 2121 of *Lecture Notes in Computer Science*, pages 3–19. Springer, 2001.

[70] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.

[71] J. Van den Bussche. Constraint databases, queries and query languages. In J. Paredaens, G. Kuper, and L. Libkin, editors, *Constraint databases*, chapter 2, pages 21–54. Springer-Verlag, 2000.

[72] L. van den Dries. *Tame Topology and O-minimal Structures*. Cambridge University Press, 1998.

[73] L. Vandeurzen, M. Gyssens, and D. Van Gucht. On query languages for linear queries definable with polynomial constraints. In E.C. Freuder, editor, *Proceedings of the 2nd International Conference on Principles and Practice of Constraint Programming*, volume 1118 of *Lecture Notes in Computer Science*, pages 468–481, Berlin, 1996. Springer-Verlag.

[74] M. Worboys. A unified model for spatial and temporal information. *Computer Journal*, 37:26–34, 1994.

Samenvatting

De periode dat gegevensbanken enkel toelieten *alfa-numerieke* gegevens te stockeren of te manipuleren ligt reeds geruime tijd achter ons. Multimedia-toepassingen vereisen gegevensbanken die beelden (zowel stilstaand als bewegend), geluid of een combinatie van beide kunnen verwerken. In deze thesis, getiteld *Tijdsafhankelijke ruimtelijke gegevens modelleren en bevragen*, concentreren we ons op het relatief jonge gebied van de *tijdruimtelijke* of *spatio-temporele* gegevensbanken of gegevensbanken die tijdsafhankelijke, ruimtelijke informatie bevatten. Dit gebied is voornamelijk geënt op de uitgebreide kennis die men in de jaren 1980–90 verwierf in het onderzoek naar ruimtelijke gegevensbanken en ook, maar in mindere mate, op kennis over temporele gegevensbanken.

Het onderzoek naar ruimtelijke gegevensbanken leidde al tot commerciële toepassingen. Populaire gegevensbanksystemen voorzien modules die toelaten ruimtelijke informatie op te slaan en te ondervragen. Geografische informatiesystemen (GIS) worden ontwikkeld voor en gebruikt in diverse toepassingsgebieden: GPS-systemen, ruimtelijke ordening, bosbeheer, het in kaart brengen van pijpleidingen, etc. In GIS zijn er twee basismethodes voor het modelleren van ruimtelijke informatie. Bij de eerste manier legt men een denkbeeldig rooster op een beeld, en wijst aan elk vakje een kleur (of een attribuut) toe. Afhankelijk van de vereiste nauwkeurigheid is het rooster fijner of grover. Een beeld wordt dan gestockeerd als een matrix van kleuren. Bij de tweede manier legt men het beeld vast door middel van de lijnen en/of krommen die de contouren van het beeld bepalen. Een beeld wordt dan bijvoorbeeld door middel van een verzameling polynomiale formules gestockeerd, die deze lijnen en/of krommen beschrijven. De gegevensbanken die ruimtelijke informatie opslaan als een verzameling formules, worden aangeduid met de term *constraint databases*. Het onderzoek in deze thesis volgt deze laatste aanpak.

In het begin van de jaren 1990 erkende men de nood aan gegevensbanken die tijdsafhankelijke, ruimtelijke informatie konden bewaren en bewerken, en niet louter statische beelden. Gegevensbanken die tijd en ruimte combineren werden aangeduid met de term *tijdruimtelijke* of *spatio-temporele* gegevensbanken of ook wel *temporele GIS*, als de nadruk ligt op het uitbreiden van geografische informatie met tijdsinformatie. We beschrijven de diverse onderzoeksinitiatieven die ondertussen werden genomen om de rol van tijd in ruimtelijke gegevensbanken te doorgronden. We geven telkens de vooropgestelde probleemstelling aan, en bespreken de voorgestelde oplossingen. Omdat ook in deze thesis een algemeen model wordt voorgesteld, beperken we ons in het overzicht tot onderzoek dat tracht tijdsafhankelijke ruimtelijke gegevens

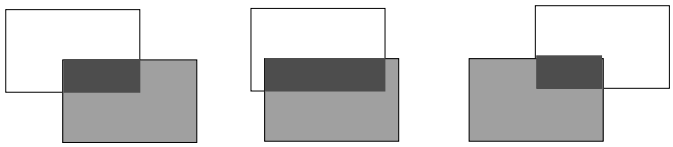
in het algemeen te modelleren, eerder dan een concreet probleem op te lossen dat de opslag van zulke gegevens vereist.

We gebruiken het constraint database model voor het modelleren van tijdsafhankelijke ruimtelijke informatie. Daarom herhalen we kort de basisprincipes van constraint databases. We leggen in het bijzonder de nadruk op het concept *genericiteit* van gegevensbankbevragingen. Een vraag om informatie die aan een gegevensbank wordt gesteld is *generisch*, als haar antwoord onafhankelijk is van de precieze manier waarop de informatie werd gestockeerd. Toegepast op ruimtelijke gegevensbanken kan dit betekenen dat de keuze van referentiestelsel of afstandsmaat (bijvoorbeeld *centimeter* versus *inch*) niet belangrijk is voor het antwoord. Formeel kan dit worden aangeduid door te eisen dat het antwoord van een vraag precies is tot op een welbepaalde transformatie van de ruimte na. We bespreken ook enkele talen die zo ontwikkeld werden dat ze enkel toelaten generische vragen te stellen aan ruimtelijke gegevensbanken.

Vervolgens onderzoeken we de betekenis van genericiteit voor vragen aan gegevensbanken die tijdsafhankelijke, ruimtelijke informatie bevatten. We stellen voorop dat het uni-directionele karakter van de tijd steeds bewaard moet blijven en stellen een aantal genericiteitsklassen voor waarin zowel temporele, ruimtelijke als meer typisch *tijdruimtelijke* aspecten als snelheid en versnelling belangrijk zijn. Als meest algemene klasse kiezen we de *tijdsafhankelijke affiene transformaties*. In *robotica*, *computer graphics* en ruimtelijke gegevensbanken werd het belang van *affiene genericiteit* reeds onderstreept door de algemeen aanvaarde *weak perspective assumption*. Deze veronderstelling gaat er vanuit dat de foto's die een observeerder, indien hij relatief ver verwijderd is van een driedimensionaal object, neemt van dat object vanuit diverse standpunten, alle gelijk zijn tot op een affiniteit van het vlak na. We geven een temporele veralgemening van deze veronderstelling: indien twee zich verplaatsende observatoren relatief ver van een, mogelijk bewegend, driedimensionaal object verwijderd, dit object filmen, zullen beide films gelijk zijn tot op een tijdsafhankelijke affiene transformatie na. Anders gezegd, men kan op elk moment, of voor elk paar *snapshots* van beide films, een affiene transformatie van het vlak vinden die de ene op de andere afbeeldt.

Gebaseerd op de voorgestelde genericiteitsklassen voor tijdruimtelijke gegevensbanken, ontwikkelen we bevragingstalen die, gegeven een bepaalde genericiteitsklasse, enkel toelaten vragen te stellen die generisch zijn voor de transformaties van die klasse. In navolging van de reeds voor ruimtelijke gegevensbanken ontwikkelde generische ondervragingstalen, zijn dit puntgebaseerde tijdsruimtelijke talen. De tijdruimtelijke gegevens worden voorgesteld als een collectie punten met een aantal ruimtelijke en één tijdscoördinaat. Later zullen we, omdat het beschrijven van gegevens als een verzameling punten heel wat wiskundig inzicht van de gebruiker vergt, talen ontwikkelen die toelaten objecten te manipuleren.

In het eerste deel van ons onderzoek stelden we de gegevens zo ruim mogelijk voor als toegelaten in het constraint databases model. In het tweede deel onderzoeken we het gedrag van tijdsafhankelijke, ruimtelijke objecten die meer overeenstemmen met het intuïtieve beeld dat men heeft van tijdsafhankelijke, ruimtelijke informatie: een (eindige) collectie bewegende objecten. Een gegevensbank bevat een eindig aantal objecten die een bepaalde *vorm* hebben, gedurende een welbepaalde periode in de tijd, hun *tijdsdomein*, bestaan en bewegen of veranderen in de tijd volgens een

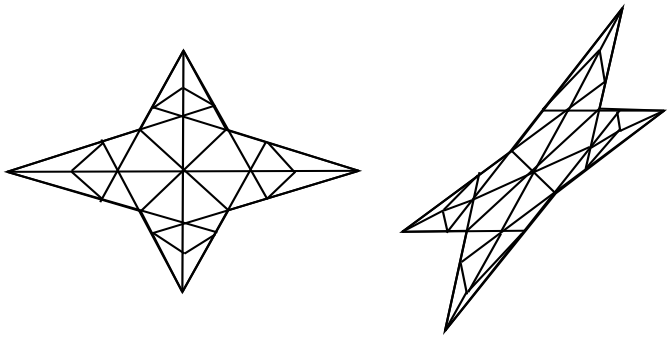


Figuur 8.2: Deze figuur toont, op drie verschillende momenten, de doorsnede van een vaste rechthoek (lichtgrijs gekleurd) en een verschuivende rechthoek (wit gekleurd). Deze doorsnede, die een scalering ondergaat, is donkergrijs gekleurd.

welbepaalde *bewegingsfunctie*. Bovendien beperken we de ruimte tot het vlak. We analyseren diverse klassen van vormen en bewegingsfuncties, en onderzoeken of de daaruit samengestelde objecten goede kandidaten zouden zijn als basisobjecten voor tijdruimtelijke gegevensbanken. Een belangrijk criterium hiervoor is *geslotenheid* onder verzameling-theoretische operaties: indien een eindige verzameling objecten tot een bepaalde klasse behoort, behoren dan ook de doorsnede, het verschil en de unie van deze objecten tot deze klasse? Figuur 8.2 illustreert dat de klasse van de verschuivende rechthoeken niet gesloten is voor de operatie doorsnede. We constateren dat slechts twee (niet-triviale) klassen van objecten voldoen aan de eigenschap geslotenheid. De eerste klasse omvat objecten die als vorm een rechthoek hebben, waarvan de zijden parallel zijn met de coordinaat-assen, en als bewegingsfunctie een tijdsafhankelijke *scaling*. De tweede en meest algemene klasse omvat objecten die een driehoek als vorm hebben, en bewegen volgens een tijdsafhankelijke affiene transformatie. We halen ook het *normalisatie probleem* aan: kunnen verzamelingen van objecten op een uniforme manier worden voorgesteld? Immers, twee verschillende verzamelingen van objecten kunnen hetzelfde fenomeen modelleren.

We onderzoeken dit probleem in detail, toegepast op de meest algemene klasse van objecten die voldoet aan de geslotenheideigenschap: de affien bewegende driehoeken. Het genericiteitsprincipe in het achterhoofd houdend, eisen we dat deze normaalvorm de gegevens slechts tot op een tijdsafhankelijke affiniteit nauwkeurig bepaalt. Aangezien de objecten bewegende driehoeken zijn, is een affien-invariante triangulatie een geschikte normalisatie. We ontwikkelen eerst een nieuw algoritme voor het affien-generisch trianguleren van een verzameling driehoeken (of polygonen) in het vlak. Figuur 8.3 illustreert de triangulatie van een stervorm en een affien beeld van deze vorm via het door ons ontworpen algoritme. In de literatuur bestond er slechts één ander zulk algoritme, gebaseerd op een affien-invariante norm. Het door ons voorgestelde algoritme is zeer intuïtief en efficiënt berekenbaar.

Deze triangulatie gebruiken we dan als basis voor een triangulatie-algoritme voor tijdruimtelijke informatie. Het resultaat van dit algoritme is een opdeling van het tijdsdomein van een verzameling objecten in intervallen, zodanig dat de schikking van de objecten *isomorph* blijft gedurende een interval in deze opdeling. Met isomorph bedoelen we dat de ruimtelijke triangulatie dezelfde structuur heeft. Het resultaat van deze tijdruimtelijke triangulatie wordt nog steeds voorgesteld door middel van ruimtelijke en tijdscoördinaten van hoekpunten. Daarom stellen we een datastructuur voor om het resultaat van ruimtelijke en tijdruimtelijke triangulaties zodanig op te slaan dat er geen exacte coördinaten meer nodig zijn. Het resultaat is een nor-



Figuur 8.3: De triangulatie van een stervorm (links) en de triangulatie van het beeld van deze ster onder affiniteit van het vlak (rechts).

maalvorm die gegevens tot op een tijdsafhankelijke affiene transformatie na bepaalt. Het bevragen van deze normaalvorm beperkt de gebruiker op natuurlijke wijze tot het stellen van affien-generische vragen.

Vervolgens grijpen we terug naar de generische punt-talen die we ontwikkelden voor tijdruimtelijke gegevensbanken. We veronderstellen dat tijdruimtelijke gegevens worden voorgesteld als een, mogelijk oneindige, verzameling vlakke driehoeken die gelijktijdig of voor of na mekaar in de tijd bestaan. Waar we voor de punt-talen konden steunen op eerder onderzoek in het gebied van de ruimtelijke gegevensbanken, is dat hier niet het geval. We ontwikkelen eerst een driehoekgebaseerde affien-generische taal voor vlakke figuren. We tonen aan dat de gevonden vlakke driehoektaal kan worden uitgebreid voor algemenere objecten in hogere dimensies. Vanuit praktische overwegingen onderzoeken we ook of we van een mogelijk oneindige verzameling driehoeken kunnen bepalen of diezelfde verzameling ook kan worden voorgesteld gebruik makend van een eindig aantal driehoeken. Het eerder ontwikkelde triangulatie algoritme voor ruimtelijke figuren helpt ons deze vraag positief te beantwoorden. Nadien stellen we driehoekgebaseerde tijdruimtelijke ondervragingstalen voor, voor de genericiteitsklasse van de tijdsafhankelijke affiene transformaties, maar ook voor de snelheid- en versnelling- bewarende transformaties, die we in het eerste deel van dit werk identificeerden.

We besluiten deze thesis met een samenvatting van de beschreven onderzoeksresultaten. Tenslotte wijzen we op enkele interessante mogelijke voortzettingen van het in deze thesis beschreven onderzoek.