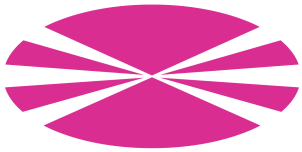

UNIVERSIDADE DA CORUÑA

DEPARTAMENTO DE COMPUTACIÓN

DUALGRID:

A CLOSED REPRESENTATION SPACE FOR CONSISTENT SPATIAL DATABASES

Tesis Doctoral
A Coruña, Septiembre 2012

Autor: José Antonio Cotelo Lema
Directores: Dr. Miguel Ángel Rodríguez Luaces
Prof. Dr. Ralf Hartmut Güting

Autor: José Antonio Cotelo Lema
Título: Dualgrid: A closed representation space for consistent spatial databases
Departamento: Computación
Directores: Dr. Miguel Ángel Rodríguez Luaces
Prof. Dr. Ralf Hartmut Güting
Año: 2012

PhD Thesis directed by:

Dr. Miguel Ángel Rodríguez Luaces
Departamento de Computación
Facultade de Informática
Universidade da Coruña
15071 A Coruña (Spain)
Tel: +34 981 167000 ext. 1254
Fax: +34 981 167160
luaces@udc.es

Prof. Dr. Ralf Hartmut Güting
LG Datenbanksysteme für neue Anwendungen
FernUniversität in Hagen
Postfach 940
D-58084 Hagen
Germany
Tel: +49 (2331) 987 4279
Fax: +49 (2331) 987 4278
rhg@fernuni-hagen.de

A Noelia, Rubén e Irene

Abstract

In the past decades, much effort has been devoted to the integration of spatial information within more traditional information systems. To support such integration, spatial data representation technology has been intensively improved, from conceptual and discrete models for data representation and query languages, to indexing and visualization technologies and interoperability standards. As a result of all these efforts, Geographic Information Systems (GIS) are nowadays a widely used technology.

The existing spatial databases technology provides standardized data models and operations [OGC06], based on conceptually solid *spatial algebras*. However, translating such conceptual models into physical models suitable for their implementation on computers, where only finite precision representations of the space can be used, becomes a difficult task. As a result, the current implementations of physical models are generally severely limited when compared to their conceptual counterparts. They attempt to provide an implementation fulfilling the original conceptual algebras, but at the physical level they cannot further ignore the problems of robustness and topological correctness arising from the use of finite precision numbers for representing spatial coordinates. This results in deceptive physical algebra implementations because they break most of the properties of the conceptual algebra that they rely on. More specifically, the physical models do not remain closed under the data types and operations of the algebra, and the solutions applied to address this problem, usually some kind of *approximated result*, do not fulfill the properties expected from the affected operation. The consequence is that the physical models fail to provide consistent implementations for the spatial operations. This makes development of applications that rely on the properties of the conceptual model (e.g., spatial analysis applications) much more complex, if not impossible. Moreover, even the implementation of the physical model itself becomes more complex, as it can not rely anymore on the theoretical basis of the conceptual model it is supposed to implement.

The main goal of this research work is to provide a framework to develop spatial database extensions capable of fulfilling the key properties of the conceptual spatial algebra they implement. At the same time, the proposed framework meets the constraints imposed by nowadays real world GIS applications in terms of performance and resource requirements, as well as interoperability with existing applications and standards.

To achieve this goal, we first analyze the current state of the art in spatial information representation. The main focus is on the way the different approaches deal with the limitations imposed by computers and the effects that these solutions have in the properties of the conceptual model they intend to implement. Second, we study the sources of these problems and propose a well-grounded physical model framework (called Dualgrid) to guarantee that the implementations of spatial algebras keep their key properties from the perspective of the user application. We also provide an example of such an implementation and experimental results on how such a framework solves the consistency and even the implementation problems of an existing and widely used spatial database extension. Third, we revisit our framework to extend its properties (DualgridFF) so that it is able to meet the additional restrictions imposed by current spatial applications, tools and interoperability standards (OGC).

Resumen

En las últimas décadas se ha dedicado un significativo esfuerzo a la integración de las tecnologías de Sistemas de Información Geográfica (SIG) con sistemas de información más tradicionales. Para dar soporte a esa integración la tecnología de representación de datos espaciales ha sido mejorada en múltiples aspectos, desde los modelos (conceptuales y discretos) de representación de datos y lenguajes de consulta a las tecnologías de indexación y visualización y a los estándares de interoperabilidad. Como resultado de estos esfuerzos, la tecnología de Sistemas de Información Geográfica es ampliamente utilizada en la actualidad en todo tipo de aplicaciones.

Las tecnologías de bases de datos espaciales actuales ofrecen modelos de datos y operaciones estandarizados [OGC06], inspirados en *álgebras espaciales* con unas bases conceptuales sólidas. En contraste, las implementaciones existentes en la actualidad sufren severas limitaciones (en comparación con los modelos conceptuales que pretenden soportar), resultantes de las dificultades inherentes a traducir esos modelos conceptuales en modelos físicos susceptibles de su implementación en ordenadores, donde es necesario usar espacios de representación de precisión finita. A pesar del esfuerzo por ofrecer implementaciones que cumplan con el álgebra conceptual original, no es posible seguir ignorando a nivel físico los problemas de robustez y corrección topológica que surgen del uso de números de precisión finita para la representación de las coordenadas espaciales. El resultado son implementaciones que sólo cumplen en apariencia con las álgebra conceptuales originales, pero que en realidad incumplen la mayor parte de las propiedades en que están basadas esas álgebras. Más específicamente, los modelos físicos no mantienen sus propiedades de cierre bajo el conjunto de tipos de datos y operaciones implementados, y las soluciones aplicadas para solventarlo, normalmente algún tipo de *resultado aproximado*, no cumplen con las propiedades esperadas de la operación en cuestión. En consecuencia, el modelo físico resultante no es capaz de ofrecer una implementación consistente de las operaciones espaciales ofrecidas a los usuarios. Como resultado, el desarrollo de aplicaciones basadas en las propiedades del modelo conceptual (por ejemplo,

aplicaciones de análisis espacial) se vuelve mucho más difícil, si no imposible. De hecho, incluso la implementación del propio modelo físico se vuelve mucho más compleja, al no poder apoyarse ni siquiera en las bases teóricas del modelo conceptual que se supone se está implementando.

El objetivo principal de esta tesis es sentar las bases para el desarrollo de extensiones de bases de datos espaciales capaces de cumplir las propiedades clave del álgebra espacial conceptual en la que se basan, teniendo en cuenta además las restricciones impuestas por la realidad de las aplicaciones GIS actuales en términos de rendimiento y consumo de recursos y de interoperabilidad con las aplicaciones y estándares existentes.

Para alcanzar dicho objetivo, se analiza primero el estado del arte actual en representación de información espacial, prestando especial atención a las limitaciones impuestas por los ordenadores y los efectos que esas soluciones tienen en el (in)cumplimiento de las propiedades del modelo conceptual. En segundo lugar, se estudian las raíces de esos problemas y se propone un marco teórico para el diseño de modelos físicos (Dualgrid) que garantiza que las implementaciones de álgebras espaciales basadas en él mantienen las propiedades clave desde el punto de vista de las aplicaciones de usuario. Como prueba de concepto, se muestra un ejemplo de una implementación basada en Dualgrid y resultados experimentales mostrando cómo su uso soluciona los problemas de consistencia y (incluso) de implementación de una extensión de bases de datos espaciales ampliamente utilizada. En tercer lugar, se revisita dicho modelo para extender sus propiedades (DualgridFF) con el fin de hacer posible el cumplimento de las restricciones adicionales (en términos de rendimiento, espacio de almacenamiento e interoperabilidad) impuestas por las aplicaciones, tecnologías GIS y estándares de interoperabilidad (OGC) existentes.

Resumo

Nas últimas décadas tense adicado un esforzo significativo á integración das tecnoloxías de Sistemas de Información Xeográfica (SIX) con sistemas de información mais tradicionais. Para dar soporte a esa integración a tecnoloxía de representación de datos espaciais ten sido mellorada en numerosos aspectos, dende os modelos (conceptuais e discretos) de representación de datos e linguaxes de procura ata as tecnoloxías de indexación e visualización e os estándares de interoperabilidade. Como resultado destes esforzos, as tecnoloxías de Sistemas de Información Xeográfica son amplamente utilizadas na actualidade en todo tipo de aplicacións.

As tecnoloxías de bases de datos espaciais actuais ofrecen modelos de datos e operacións estandarizados [OGC-SFS], inspirados en álxebras espaciais con unhas bases conceptuais sólidas. Por contra, as implementacións existentes na actualidade sofren severas limitacións (en comparación cos modelos conceptuais que pretenden soportar), resultantes das dificultades inherentes a traducir eses modelos conceptuais en modelos físicos susceptíbeis da súa implementación en ordenadores, onde é preciso usar espazos de representación de precisión finita. A pesar dos esforzos por ofrecer implementacións que cumpran coas álxebras conceptuais orixinais, non é posible seguir ignorando a nivel físico os problemas de robustez e corrección topolóxica que xorden do uso de números de precisión finita para a representación das coordenadas espaciais. O resultado son implementacións que sómente cumpren en aparencia coas álxebra conceptuais orixinais, pero que en realidade incumpren a maior parte das propiedades en que están baseadas esas álxebras. Mais especificamente, os modelos físicos non manteñen as súas propiedades de peche baixo o conxunto de tipos de datos e operacións implementados, e as solucións aplicadas para solventalo, normalmente algún tipo de *resultado aproximado*, non cumpren coas propiedades esperadas da operación en cuestión. En consecuencia, o modelo físico resultante no é capaz de ofrecer unha implementación consistente das operacións espaciais ofrecidas aos usuarios. Como resultado, o desenvolvemento de aplicacións baseadas nas propiedades do modelo conceptual (por exemplo, aplicacións de análise espacial) tornase moito

mais difícil, se non imposible. De feito, incluso a implementación do propio modelo físico se fai moito mais complexa, ao non poder apoiarse nin sequera nas bases teóricas do modelo conceptual que se supón se está implementando.

O obxectivo principal de esta teses é sentar as bases para o desenvolvemento de extensións de bases de datos espaciais capaces de cumprir coas propiedades clave da álxebra espacial conceptual na que se basean, tendo en conta ademais as restricións impostas por a realidade das aplicacións SIX actuais en termos de rendemento e consumo de recursos e de interoperabilidade coas aplicacións e estándares existentes.

Para acadar o devandito obxectivo, analizase primeiro o estado da arte actual en representación de información espacial, prestando especial atención as limitacións impostas por os ordenadores e os efectos que esas solucións teñen no (in)cumprimento das propiedades do modelo conceptual. En segundo lugar, estúdanse as raices de eses problemas e proponse un marco teórico para o deseño de modelos físicos (Dualgrid) que garante que as implementacións de álxebras espaciais baseadas en el manteñen as propiedades clave dende o punto de vista das aplicacións do usuario. Como proba de concepto, amosase un exemplo de unha implementación baseada en Dualgrid e resultados experimentais mostrando como o seu uso soluciona os problemas de consistencia e (incluso) de implementación de unha extensión de bases de datos espaciais amplamente utilizada. En terceiro lugar, revisítase o devandito modelo para estender as súas propiedades (DualgridFF) coa fin de facer posible o cumprimento das restricións adicionais (en termos de rendemento, espazo de almacenamento e interoperabilidade) impostas por as aplicacións, tecnoloxías SIX e estándares de interoperabilidade (OGC) existentes.

Acknowledgements

I would like to thank all those who, directly or indirectly, have helped this thesis come to be written. Especially, to Ralf Hartmut Güting, Nieves Rodríguez Brisaboa, Miguel Rodríguez Luaces, Roberto Creo Hombre, Miguel Rodríguez Penabad and my family.

Also, I would like to thank the *Databases Lab* at University of A Coruña, the *Datenbanksysteme für neue Anwendungen* group at FernUniversität Hagen and the CHOROCHRONOS project. Had they not existed, this thesis would not have existed either.

Agradecimientos

Mis agradecimientos a todos aquellos que, directa o indirectamente, han ayudado a que esta tesis llegase a ser escrita. En especial, a Ralf Hartmut Güting, Nieves Rodríguez Brisaboa, Miguel Rodríguez Luaces, Roberto Creo Hombre, Miguel Rodríguez Penabad y mi familia.

Igualmente, al Laboratorio de Bases de Datos de la Universidad de A Coruña, al grupo *Datenbanksysteme für neue Anwendungen* de la FernUniversität Hagen y al proyecto CHOROCHRONOS. Si no hubiesen existido, esta tesis tampoco lo habría hecho.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

This thesis presents the research and development work performed in the area of spatial databases towards the definition of a physical data model that keeps the theoretical properties of conceptual models.

To achieve this goal, we study the capability of existing spatial physical models to reliably translate all the semantics and properties of their original spatial conceptual models to the real world applications and spatial database extensions. Then, we analyze the consequences of their failure to keep the conceptual model properties. After that, we propose a new perspective to develop physical representation models that succeed in translating the key conceptual model properties to the final implementations. Our physical model drastically eases the use of such spatial database implementations by GIS application developers. It also increases their capabilities to develop applications with spatial reasoning functionalities.

1.1 Background and motivation

There has been much research effort in the last two decades on representing, querying and exchanging spatial information in geographic information systems (GIS) and spatial database management systems (SDBMS). This effort lead these technologies to evolve from the original ad-hoc geographic-only applications (that were common in the early 90s) to the nowadays standardized, interoperable and widely-used technologies that enable applications in different domains to integrate spatial and non-spatial information.

The improvement of the GIS field is mainly driven by advances in spatial information modeling and technologies, which enable database management systems and applications to take advantage of spatial information. The use of correct spatial information models is fundamental for the evolution of the field, therefore, an important research effort has been devoted to it.

The process of representing geographic information in a computer involves the definition of a set of *data models* to represent real-world geographic features using appropriate data structures. Typically, three different data models defined at three different abstraction levels are used: the *abstract spatial model* (also called *conceptual model*), the *discrete spatial model* (also called *logical model*) and the *physical spatial model*.

The abstract spatial model describes the geographic features of the real world, the relationships between different features and the operations that can be performed to each feature. It uses formal concepts defined without taking into consideration the implementation details. An example of an abstract spatial model is the ISO 19107:2003 international standard [ISO03], which defines conceptual types (e.g., *curve*, *surface*) and operations (e.g., *overlaps*, *touches*) for geographic objects.

The discrete spatial model takes into account the limitations of a computer system (e.g., limited memory and computing performance) to define data types and algorithms that can be used to implement the concepts of an abstract data model. The Open Geospatial Consortium standard for Simple Features in SQL [OGC06] is a common example of a discrete spatial model for ISO 19107:2003. This model defines data types such as *linestring* and *polygon* and ensure that the data types specifications are appropriate to support the implementation of efficient algorithms (with regard to the their algorithmic complexity) for the operations it defines.

Finally, a physical spatial model is a particular implementation of a logical model in a specific computing environment. For example, the implementation of OGC SFS in PostGIS [Ref10] for PostgreSQL using GEOS is a physical data model. A physical model also defines some aspects left open by discrete models, like the particular representation of spatial coordinates and the precision used for it.

Abstract spatial models define an algebra of spatial data types and operations with solid theoretical properties. They ensure that these operations are closed, that is, that the data types are powerful enough to represent any result of the spatial operations (e.g., an abstract spatial model must ensure that the result of the *intersection* of any two *surface* values can be represented using a *surface* value). Regarding the underlying geographic space, they assume a continuous space over $\mathbb{R}^3$ [ISO03].

There has been much work over the last decades on the definition of abstract spatial models, and these definitions are so mature that international standards like the aforementioned ISO 19107:2003 have been defined and approved. However, the

definition of discrete and physical spatial models that maintain the properties of an abstract spatial data model has proven to be a difficult problem.

Discrete spatial models must define data types using a finite number of components, and the designer must decide whether a computer-friendly representation is used or not. For example, a discrete data model may represent curves using a set of linear segments, surfaces using a set of segments to represent their boundary, and points with a pair of numeric coordinates. If the discrete spatial model continues to assume a continuous space, it can claim that the properties of the abstract spatial model are maintained, but the problem of representing the spatial coordinates in a computer (the discretization of the geographic space) is translated to the physical spatial model. However, if the problem is addressed at the discrete spatial model by using a finite space for coordinates (e.g., 64 bits integers or IEEE754 double-precision floating-point numbers), then it is really difficult to maintain the properties of the abstract spatial model. For example, it is not possible to represent the point with coordinates $(\frac{1}{3}, \frac{2}{3})$ using double-precision floating-point numbers. Furthermore, if two segments (e.g., $S_1 = ((0,0),(1,2))$ and $S_2 = ((0,1),(1,0))$) intersect at these coordinates, the point resulting from the intersection operation cannot be represented precisely, and subsequent predicates checking whether the point belongs to the original segments will return *false*.

Spatial databases researchers have proposed several ways of addressing the space discretization problem while trying to comply with the original spatial model properties. Two such proposals are Realms [GS93] and the use of arbitrary precision rational numbers [WS05].

In the Realms physical spatial model [GS93], each time a new spatial object is inserted into the database, all boundary intersections with the existing objects are detected. The objects involved are rewritten so that such intersection points are made explicit in their representation. This ensures that, even if an intersection point needs to be approximated to the underlying finite resolution space, all the involved objects are adjusted so that such point continues to be part of their boundary. The result is that, once the object is inserted in the database, all the spatial operations of the abstract spatial model implemented by their proposal (the ROSE algebra [GS95]) remain closed and coherent among them. However, this coherence is only maintained as far as no new external objects are inserted (other than the ones resulting of spatial operations over the already existing objects). Therefore, consistency is not guaranteed between answers before and after an insertion. Moreover, its implementation in commercial[1] databases can be problematic, because existing spatial objects can get modified without

[1] In the following, we will use the term *commercial* to refer to final product implementations in contrast to research/experimental implementations, regardless of their business model (open source or proprietary software).

user knowledge. In fact, the model implies that spatial object insertions made by a user will cause modifications in objects over which the user may have no update (or even read) privileges.

Some authors [WS05] propose the use of arbitrary precision rational numbers as the underlying coordinates space. Although this would guarantee that the properties of the original model are maintained, it imposes a big toll in performance and storage requirements [BF09]. Therefore, no (commercial) spatial technology or DBMS has adopted it.

Following a different approach, spatial database technology implementors (Geomedia, ESRI, PostGIS, etc.) have systematically assumed that space discretization problems were inherent to the discretization process and that adopting hardware-supported representations for coordinates is a requirement (that is, integer or floating-point numbers). The first implementations of spatial DBMS usually removed any operation of the abstract spatial model that did not remain strictly closed under the coordinate space used in the discrete model. However, most customers preferred to have operations returning an approximation of the theoretical result (that could be represented with the given data types) rather than not having the operation at all. Therefore, current spatial DBMS implement the full set of operations of the abstract models, using *approximate versions* of the problematic ones (e.g., an *intersection* operation that, to be fair, should be called *approximate_intersection*). With the adoption of *approximate operations*, they have focused on solving their own implementation issues. For example, they usually ensure that their approximation is still a valid spatial value after the approximation, so it can be used as the input for another operator. They provide exact spatial predicates (implemented with algorithms such as [ABD+97]) that guarantee that all precision issues are properly handled to provide the correct answer for all spatial operations returning boolean values. They also ensure that their implementation of more complex operations takes the correct decisions. But, once they have addressed their problems, the remaining ones are left to their users.

As shown, the development of physical spatial models that properly solve the space discretization problems in a way suitable to be implemented in commercial DBMS remains, despite the research efforts, unsolved. Moreover, the evolution of spatial technologies is not driven by a coordinated effort. Instead, the improvements in spatial technologies are being driven by different forces corresponding with the (usually not coincident) perspectives of four collectives: spatial database researchers, commercial spatial database technology[2] developers, GIS software developers, and GIS application

[2]We will use the term *commercial spatial database technology* to refer to Spatial DBMS as well as other technology for data management not directly related to databases, such as programming libraries and GIS

users. In the following paragraphs we describe the points of view and motivations of each of these four collectives. The failure to conciliate the perspectives of these four collectives, combined with the existing open problems, undermines the evolution of the GIS field.

Spatial database researchers tend to focus on proposing well-defined and powerful theoretical models, without taking into account how well they can be adapted to commercial DBMS requirements.

Commercial spatial database technology developers focus on creating database management systems, libraries and tools. However, they are still not oriented to final users, but rather to software developers. Therefore, they need to achieve a trade-off between a robust model implementation, an appropriate performance, low storage requirements, and power of their implementations. In addition to that, getting a solid implementation is a key requirement. Hence, if they can manage to get the problem sorted out well enough to get their code working without a sensible penalty in the rest of aspects, it is enough for them. For the remaining part of the problem, they can try to pass it on to the following element of the chain: GIS software developers. Of course GIS software developers would prefer solutions that solve the problem and avoid them any headache, but that would only make a difference in the selection of spatial technologies if there were significant differences between solutions regarding this issue, and not just "different flavors of tricks".

GIS software developers need to provide final users with the functionalities they require. They can hardly ignore the problem or move it to the user, because their software is the one expected to do spatial reasoning. For example, if a user just needs to know which pieces of land are adjacent to a given one, we could draw a map of the area around the given piece of land and let him select the adjacent ones. This way, the application itself does not need to implement the detection method. But if the application itself needs such information to compute the result the user needs (for example, to select the pieces of land adjacent to the ones that meet some criteria), then it should find it out by itself, and can not "pass the problem" on to the user. Therefore, GIS software developers would like to use spatial technologies that do exactly what they are expected, and that comply, as much as possible, with the original abstract model (which was, in fact, designed having the functionality requirements of GIS software developers in mind). The less the spatial technologies comply with the abstract model, the bigger the problems they will have when they try to get their applications working. Just for a moment, think of implementing a simple small program based in a programming language that has approximate *boolean operations*

development tools. They are grouped together because their developers share a common perspective with regard to spatial models that plays, in fact, a big role in their common priorities.

(which 1 in 100 times return *false* for $a = a$), approximate *counters* (you can try to implement a loop using floating-point-based counters) or *approximate ifs* (which in fact are the result of an *if* with an approximate boolean operation). It would probably be funny, as long as you do not have the client waiting for it and your boss accounting the hours.

Finally, GIS application users (as any application user) need the applications to provide them with the functionalities they require. The implementation details are irrelevant as long as they have an acceptable performance, they behave as expected, and they get the work done. GIS application users will usually be non-expert users in spatial technologies, and they need the applications to behave in a predictable and intuitive way.

Current commercial spatial DBMS/technology implementations break the main properties of the abstract data model, either because they implement an ill-defined discrete data model, or because they had to address somehow the space discretization needs that the discrete data model avoided to address. As a result, their users (GIS software developers) can no longer rely on such properties when developing their applications. This is not a minor problem, because the abstract data model properties had been carefully chosen to model the reality. If the basic properties are not maintained, the simplest spatial reasoning algorithms become difficult to implement. For instance, suppose that a basic set-theory property is not maintained due to closure problems between data types and operations, which is fairly common in commercial spatial database technology implementations. If the intersection point of two segments $P = S_1 \cap S_2$ cannot always be represented, and hence we just represent an approximation P', it turns out that when we need to check which segments P' belongs to, we could get that neither $S1$ nor S_2 contain it. Similarly, if no consistency is provided among operations, it is possible to get *false* when testing whether $A \cap B \in A$[3]. With these incoherencies, it is really difficult to implement any spatial reasoning support in GIS applications. As a result, the implementation of spatial reasoning functionality tends to be unusual in GIS applications.

This thesis focuses on covering the gap that exists between the research work on discrete spatial models and the implementations in commercial spatial database management systems. First, it analyzes the properties that a physical spatial data model needs to fulfill to be able to implement a discrete spatial data model without breaking its closure properties, that is, to ensure that the result of any operation of the discrete model continues to be representable at the physical model. Second, it proposes two new physical spatial data models, called Dualgrid and Dualgrid For Floats (DualgridFF),

[3]Some of the current commercial spatial DBMS/technology implementations (e.g., Geomedia by Intergraph) try at least to do a small effort and, for example, for any two segments s_1 and s_2 the test $s_1 \cap s_2 \in s_1$returns always *true*.

which succeed in maintaining the closure properties at the physical level. Dualgrid defines a finite resolution representation space that guarantees all the properties required to keep the discrete data model closed. DualgridFF goes a step forward, meeting the additional requirements of real commercial databases and technologies. DualgridFF represents a trade-off between the needs of spatial database researchers, commercial spatial database technology developers, GIS software developers and GIS application users. It provides a solid physical representation model that succeeds in keeping the main properties of the original abstract and discrete vectorial models. This way, spatial DBMS and technology implementors can easily translate them to their implementations whereas the behavior of the operations is kept intuitive and well-defined. This should allow application developers to focus on the problems they have to solve (the applications). At the same time, its design makes it possible to implement them without expensive performance and storage costs. It also illustrates how important is to have into account all stakeholders[4] needs when designing physical data models.

1.2 Goals

The problems presented in the previous section motivated the primary goal of this thesis: to improve the applicability of research works in discrete spatial models to commercial databases and technologies, allowing technology developers to avoid dealing with spatial model closure problems. This will improve the applicability of those technologies and spread the use of spatially enabled applications with spatial reasoning capabilities.

To achieve this, we need to reach the following specific goals:

- Analyze the problems that are generated when a discrete/physical spatial data model breaks the original abstract model properties.
- Identify the issues that have to be addressed by the physical spatial data model to keep the properties of the discrete and abstract data models.
- Propose a first physical spatial data model that allows to recover the properties of the abstract spatial model that were lost in the implementation, while at the

[4]The term stakeholders is used in business to refer to all parts which have some impact from/to business operations. In this case, we use it to refer to all the parts that are affected by the decisions taken at the physical model. As it happens with business stakeholders, physical model stakeholders needs should be taken into account, as the positive and negative impacts that design decisions could take on their needs will drive their own decisions. The different stakeholders interests not being aligned will jeopardize the possibilities of the proposed model to have a real impact in final users.

same time trying to require as few changes as possible in the data structures and algorithms that are already implemented.

- Propose a second physical spatial data model that allows to keep the properties of the abstract spatial model and at the same time can be efficiently implemented in new commercial spatial databases and technologies, or efficiently incorporated into existing ones through more extensive changes in their data structures and algorithms.

1.3 Scope and relevance

Spatial databases technology has reached a high level of maturity. Nowadays all relevant database management systems (PostgreSQL, MySQL, Oracle, DB2, Microsoft SQL Server, Informix, etc.) provide spatial data types and operations, usually implementing widely accepted spatial standards (SFS, ISO SQL/MM, etc.). However, all existing implementations suffer a common base problem: they are not robust/consistent, in the sense that they fail to fulfill even the more basic theoretical properties of the abstract model they intend to implement.

This thesis makes three main contributions to the field:

1. It establishes the basic properties that a physical spatial model needs to meet to ensure that it is able to correctly implement a vectorial discrete model without breaking its theoretical properties.

2. It defines the Dualgrid representation space, designed to reincorporate to existing spatial implementations (e.g., ROSE algebra, PostGIS/GEOS, etc.) those robustness/consistency properties originally lost when implementing the discrete model.

3. It defines the DualgridFF physical spatial model, aimed at the implementation of new spatial databases and technologies. It provides all the benefits of Dualgrid while allowing the implementation of commercial quality spatial models, in terms of performance, storage overload and interoperability.

That is, the contributions of this thesis allow a qualitative improvement on spatial databases/technologies that should provide developers of spatially-enabled applications with the solid grounds they are demanding.

1.4 Thesis outline

Chapter 2 studies the current state of the art in geographical information systems and spatial representation. It outlines the requirements of modern geographical information systems, and gives an introduction to the more common spatial representation models. Although it focuses in vectorial representations, other representation models (constraint databases, raster, etc.) are presented. For each of them, the chapter describes how physical models address the discretization problems, as well as the impact that such decisions have from the user applications perspective.

Chapter 3 analyzes in detail the spatial operations consistency provided by the more common physical representation models.

Chapter 4 focuses on the key reasons for such inconsistencies and proposes a new physical model framework to provide spatial databases implementations that fulfill the requirements of their conceptual model counterpart.

Chapter 5 goes a step forward, and revisits the proposed physical model to incorporate the restrictions imposed by current commercial GIS applications and standards, so that the proposed physical model is suitable for its use in commercial spatial databases and technologies.

Finally, Chapter 6 concludes the thesis and points to future research directions.

Chapter 2

State of the art

This section shows an overview of the state of the art in geographic information systems (GIS) and spatial data representation. It gives an introduction to the GIS field, and more specifically to the more relevant spatial representation models. The section also shows how the different spatial physical models address the discretization of the space (i.e., the representation of spatial coordinates using finite-size representations) and how they handle the problems that arise from it. Furthermore, it also highlights the impact of those decisions from the perspective of the user applications. Finally, this state of the art focuses mainly in vectorial representation models, as they are widely used in GIS applications, which are specially affected by the space discretization problems.

2.1 Geographic Information Systems and Spatial Information Modeling

The exponential improvement in the performance of computer systems and the advances in spatial information modeling in the last decades have made possible the appearance of new tools (*Geographic Information Systems*, GIS for short) to manipulate the geographic properties of objects and to represent them in a graphical way as a map on a computer screen.

Geographic Information Systems are more than just cartographic tools to produce maps because they are a step forward over traditional information systems. They offer an appropriated environment for capturing, storing and managing both alphanumeric and geographic information, and they provide tools for processing and analyzing them together. By geographic information we mean here information about the spatial

properties of objects. This information can be as simple as the position in the map of all the hospitals of a country or as complex as the partition of the country's land with regard to the kind of vegetation that grows on it.

According to [BM98], any GIS application should provide certain functionalities that can be classified as follows:

1. *Data input and verification*. This covers all aspects of capturing and verifying the correctness of geographic data, as well as their conversion to digital form.

2. *Data storage and management*. This functionality deals with all the aspects related to the structure and organization of geographic information. It must take into account both the way the geographic information is perceived by the users (abstract model) and the way it is handled by the computers (discrete and physical models).

3. *Data transformation and analysis*. This functionality is covered by the processes for editing the information (to keep it up to date or to remove errors) and for analyzing it. Data analysis is one of the main tasks of GIS, and it consists in the application of analysis methods to the information to achieve answers to the questions posed by the users.

4. *Data output and presentation*. The functionality of producing maps and map-based material is a highly distinctive feature of GIS compared with a general purpose information system. Together with the analysis techniques, this is the aspect that differs the most from traditional information systems.

A GIS must allow the efficient exploitation of all the information it manages, not only providing spatial operations for geographic data, but also allowing to analyze and browse these data graphically, and allowing the identification of geographic relationships between objects. Examples of application domains for GIS are, among others, cadastre management, sanitation and communication networks, computer assisted navigation, and decision support systems. Some of them have even become highly popular among home users in the last decade:

- Online mapping services and applications as *Google Maps* (`http://maps.google.es`), Microsoft *Bing Maps* (`http://www.bing.com/maps/`) or *Yahoo! Maps* (`http://maps.yahoo.com/`) have popularized the access to worldwide maps with a detail never dreamed a decade ago.

- GPS Navigation software (to be run on dedicated GPS devices and/or in modern general-purpose smartphones with GPS support) has become a

common tool for route planning, for both commercial and domestic daily use. *TOMTOM* (http://www.tomtom.com/), *Navman* (http://www.navman.com/), *ROUTE66* (http://www.66.com/route66/) and *iGo* (http://www.igomyway.com/) are a few examples.

- Content geolocation is starting to become common. Examples are the geolocation-oriented photo sharing services *Flickr* (http://www.flickr.com/) and *Panoramio* (http://www.panoramio.com/), or the persons geolocation services *Google Latitude* (http://www.google.com/mobile/latitude/) and *Foursquare* (http://www.foursquare.com/).

It is important to distinguish clearly between a GIS application and a tool to develop GIS applications (herein GIS development tool). GIS development tools provide developers with the capabilities required for capturing, storing and managing both alphanumeric and geographic information. They are used as the basis to develop GIS applications that provide users with an environment adapted to their specific needs. This difference is somehow similar to the one between database management systems (or software development tools) and information systems.

Examples of GIS development tools are:

- Spatial database management systems: database extensions to provide support for spatial information. Examples of this type are Oracle Spatial (http://www.oracle.com/es/products/database/options/spatial/index.html), the commercial spatial extension for Oracle, and PostGIS (http://www.postgis.org/), a spatial extension for PostgreSQL, released as open source software.

- Geospatial web services: services for spatial data publishing, usually following OGC standards as WFS [OGC09], WMS [OGC06b] or WCS [OGC09b]. Examples are the open source geospatial servers GeoServer (http://www.geoserver.org/) and MapServer (http://www.mapserver.org/).

- GIS desktop clients, as the open source GIS clients QGIS (http://www.qgis.org/) and uDIG (http://udig.refractions.net/).

- Libraries and APIs for online spatial data visualization, as the open source JavaScript library OpenLayers (http://www.openlayers.org/) or the Google Maps API (http://code.google.com/apis/maps/index.html).

- Wide range GIS development platforms, trying to cover a wide range of GIS development aspects. Example are the commercial solutions provided by ESRI (http://www.esri.com/) and Intergraph (http://www.intergraph.com/).

The popularization of GIS in the last decade is illustrated by recent initiatives to promote the interoperability of public data infrastructures and the access to free and public spatial data sources. Two of these initiatives are INSPIRE and OpenStreetMap.

- INSPIRE (`http://inspire.jrc.ec.europa.eu/`) is an European Directive (Directive 2007/2/CE) whose goal is to establish an infrastructure for spatial information in Europe to support Community environmental policies. INSPIRE forces the creation of public spatial data infrastructures and initially addresses the publication of 34 spatial data themes needed for environmental applications. The public spatial data infrastructures publish the information following widely used OGC spatial standards.

- OpenStreetMap (`http://www.openstreetmap.org/`) is a collaborative project to create a free editable map of the world. The maps are created using data from portable GPS devices, aerial photography, other free sources or simply from local knowledge. Both rendered images and the source vector dataset are available for download under a Creative Commons Attribution-ShareAlike 2.0 license.

Both initiatives emphasize the needs of spatial data interchange, highlighting the importance of semantic and technical interoperability of spatial data and tools.

The area of GIS has been widely covered by some authors from different perspectives. [RSV01] helps researchers new to the field to get a wide understanding of the current state of the art in spatial databases technologies, from spatial data modeling and representation to storage, retrieval and manipulation, as well as algorithms and indexing methods. [Wor04], on the other hand, is a good reference for computer science professionals new to the development of spatial and GIS technologies, as it addresses GIS from a computing perspective. For GIS users, [BM98] becomes a good reference to understand the applications of GIS, providing a wide perspective of the field and giving an introduction to the theoretical and technical principles that need to be understood to work effectively and critically with GIS.

The completely different perspectives of these three books proves the existence of different interest groups in the GIS field, such as those described in Chapter 1 (spatial database researchers, commercial spatial database technology developers, GIS software developers, and GIS application users). Each of these groups has a different perspective on what is highly important, what is almost irrelevant, and what is someone else's problem.

A remarkable characteristic of current GIS systems is their high interoperability requirements. The high costs of producing spatial data, the importance of data analysis and processing to convert those data in information relevant to users with completely different needs, and the advantages of spatial information sharing among organizations

have made interoperability and modularity a key aspect in the evolution and growth of the GIS field.

Nowadays, a typical Geographical Information System relies in spatial database systems to store its data. Those data have been generated by several different sources (using their own specific GIS applications) and preprocessed when fed to the system to fit to the organization requirements. Some input data are processed automatically whereas other are introduced and managed by specific GIS applications. The information is processed again to create different presentations that fit the specific view of the world of different user types. Some information is even provided through standardized interfaces (SFS [OGC06], WFS [OGC09], WMS [OGC06b], WCS [OGC09b], etc.) that can used by other organizations for different applications.

To be able to succeed in such a complex environment, the GIS domain has drastically evolved from the ad-hoc systems in the early nineties to the nowadays highly standardized, interoperable and modularized systems. The wide effort in standards definition, mainly driven by the Open Geospatial Consortium and ISO, in data representation (SFS [OGC06], GML [OGC07, ISO07b], ISO 19107:2003 [ISO03], etc.), functional modules (ISO 19142:2010 [ISO10], WFS [OGC09], WMS [OGC06b], WCS [OGC09b], etc.) and even metadata (CSW [OGC07b]) and processing (WPS [OGC05]) services, and their adoption in the GIS field, has been a key factor on this evolution. Both OGC and ISO families of GIS services standards follow a modular approach, where each specific standard addresses some specific functionality requirements. This approach simplifies the development of GIS development tools, as developers can focus at each moment on one specific group of functionalities. It also promotes and ensures interoperability between implementations, as service modules will interconnect through well know standards.

However, previous to all this evolution, the development of generic spatial data models (initially as a research field on its own [Güt88, SH91, GS95, Ege94] and later in the form of international standards [ISO03, ISO05a, ISO05b, OGC06]), has been of fundamental importance. Such data models pursue two basic goals. On the one hand, they provide a set of spatial data types suitable to accurately and efficiently represent the kind of spatial data managed by a wide set of application environments. On the other hand, they offer a powerful set of operations over these types that fulfills the requirements of those applications.

Spatial data models are usually classified as abstract (sometimes called conceptual), discrete (sometimes called logical) or physical spatial data models, depending on the abstraction level at which they are defined. Abstract spatial data models focus on the definition of conceptually meaningful representations of real world spatial information, as well as a powerful and meaningful set of operations for exploiting it. They are designed trying to model the spatial information in a way similar to how

it is understood, classified and analyzed by GIS application users. Discrete spatial data models try to adapt abstract data models to the reality of computers and algorithm complexity, where finite storage space and computation power need to be taken into account. Finally, physical data models map discrete spatial data models to specific data structures and algorithms so that they can be directly (and efficiently) implemented.

In the following sections each of these three types of spatial models are explained in more detail.

2.2 Abstract spatial models

Research proposals first ([GS95, LTR99]) and international standards later ([ISO05a, ISO03, ISO05b]) have defined abstract spatial data models attempting to capture the semantics of data types and operations as they are seen by spatial information users, settling a formal and high level basis to represent and query spatial information. They represent spatial information over a continuous *geographic space*, a space of coordinates (usually $\mathbb{R}^2$ or $\mathbb{R}^3$) over which spatial data are mapped. This space is, usually, either Cartesian (the GIS uses a flat model of the earth) or geodesic (the space tries to better model the reality by taking into account the Earth's shape and curvature, trying to represent distances and areas in a more realistic way).

Current international standard ISO 19109:2005 [ISO05a] defines the way GIS applications should model information representing the real world through *features*. A *feature* describes objects with a geographic location (buildings, a digital terrain model, a map, etc.). The standard formalizes the *features* structure through the *General Feature Model* (GFM). In it, *features* have a type (e.g., roads, rivers, buildings, etc.), attributes, relations between feature types and behaviors. *Features* can be either *geographic objects* or *coverages* (space mapping functions). *Geographic objects* are defined in depth at abstract level by the standard ISO 19107:2003 [ISO03], whereas *coverages* are specified in standard ISO 19123:2005 [ISO05b].

According to ISO 19107:2003 [ISO03], a *geographic object or entity* is an application object for which the GIS stores geographic attributes and (optionally) alphanumeric attributes. A *geographic attribute* represents a geographic property (position, extension, etc.) of an object. A *geographic domain* is the set of values that a *geographic attribute* may have. The more relevant *geographic domains*, according to ISO 19107:2003 [ISO03], are:

- *Primitives*: basic types. They can be:
  - *Point*: represents a single point in the space, for example the location of a farm.

  - *Curve*: represents a sequence of contiguous points in the space (a curve). An example of such value is the course of a road.

  - *Surface*: represents a connected area in the space, possibly with holes. A piece of land or the area belonging to a municipality are examples of such values.

- *Complex*: they are a combination of *primitive* elements representing a single object. They might be either homogeneous, where all the elements belong to the same type (called *composites*), or heterogeneous, where elements of various types coexist. An example of the former is the area covered by the snow in a country. The hydrography of a given area, including both rivers (lines) and lakes (regions) is an example of the latter.

- *Aggregates*: collections of *primitive* elements. They differ from *complex* in that they intend to represent a collection of objects, instead of a complex object.

Coverages are specified by the standard ISO 19123:2005 [ISO05b] as a data representation that directly assigns values to geographical positions. A *coverage* is a function from the geographic domain to a value of other domain (numeric, classification, etc.), where each geographic location has a unique value assigned. Coverages can represent both discrete or continuous functions. The standard defines several *interfaces* for different types of mapping methods (discrete coverages, rectangular or hexagonal grids, Triangulated Irregular Networks (TIN), etc.). But, as it corresponds to an abstract model, it does not impose any limitations on how the data are stored or managed.

Figure 2.1 shows two possible representations of a set of features, depending on the information relevant for users at two different scales. At scale E1 the object *c* is represented using the entity *city_locations*, whose geographical location is represented by an attribute of type *point*. The geographic object *r* belongs to a feature *rivers* whose geographic attribute is represented by an attribute of type *line*. For scale E2, cities are represented by the entity *city_areas*, whose geographic attribute is a *region*. The *rivers* feature is also represented. In addition, coverages *vegetation* (a *discrete coverage*, splitting the space into the vegetation types *ts1*, *ts2*, *ts3*, *ts4*) and *salinity* (a *continuous coverage*, represented in the map with darker colors to represent lower soil salinity) are also depicted at scale E2.

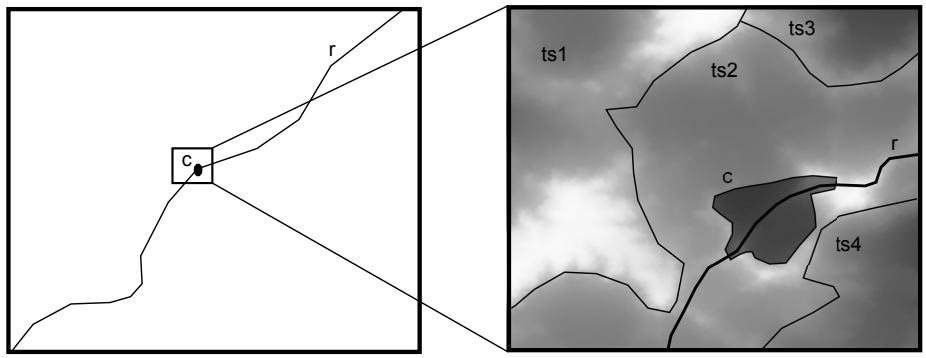

Figure 2.1: Examples of geographic data represented at two different scales.

2.3 Discrete spatial models

Conceptually, the spatial values represented by abstract models are usually non-empty and infinite subsets of the geographic space. However, in order to represent such values in a computer system (and in order to implement most operations over them), infinite sets must be modeled by some kind of finite representation.

Several discrete spatial data models have been proposed in the spatial scientific literature, some of them being nowadays widely used. They try to provide efficient and finite representations for the usually infinite sets of points of spatial objects. These models can be roughly classified into a few basic categories [Par95] depending on the approach followed in the representation and the intended target application domains. The three main categories are the following:

1. *Raster models*. Raster models are based on the concept of *maps of bits*. The infinite points of the space are represented by a finite number of raster points, which are uniformly distributed over the space. They are usually represented using some kind of array data structure. Raster representations have some advantages (typical spatial operations are intuitive and simple to implement), but have also important drawbacks, such as the big space requirements for storing spatial objects (all the points belonging to the object have to be explicitly represented). These models are often used for spatial information generated by imagery techniques, and as a way to represent *coverages*.

2. *Vectorial models*. In vectorial models [LT92], the information in the n-dimensional space is represented using m-dimensional hyperspaces, with $m < n$. More informally explained, the infinite set of points belonging to a spatial object is represented by its boundary, usually using linear representations. For instance, in the two dimensional space the following types are usually defined [GS95, OGC06]:

   - *Points*.
   - *Graphs*, composed by nodes (points) and arcs between nodes (segments).
   - *Polylines*, represented as a finite sequence of points.
   - *Polygons*, represented as a closed and no self-intersecting polyline.
   - *Complex objects*, as for example sets of polylines, complex polygons (composed by a set of polygons, possibly with holes) or heterogeneous sets of objects containing elements belonging to any of the previous types.

   Vectorial data models are widely used and numerous query languages [Cha94, Rig94, Güt94a, Ege94] and algebras [Güt88, Güt89, Güt94b] are based on them. One of the advantages of this family of models is the existence of efficient data structures [Gun88], as well as very efficient algorithms for detecting relationships between the objects [Sam90] and for computing set-theory operations. Furthermore, they are very appropriate for GIS applications where spatial objects have clearly defined boundaries (e.g., territory administration) and for visualization in user interfaces because the objects support zooms, rotations and other transformations without loosing quality. As disadvantages, using these models to represent *continuous coverages* would require to discretize and vectorize them in geometries, setting sharp and precise value boundaries to objects that conceptually are continuous. Moreover, they are not appropriate for imagery representation.

3. *Constraint databases models*. Under the constraint databases models, data are represented using linear constraints. The same approach can be used for representing spatial data in the space, as shown in [KPV95, GK97, GRSS97, GRS98]. For example, the constraint $x > 1 \wedge 2x - y - 5 < 0 \wedge y < 7$ (displayed in Figure 2.2.c) represents a region in the space. The advantage of this approach is that the extension of constraint databases systems to the management of spatial information is quite straightforward, representing them through constraints, similarly to all other data types. The disadvantages are the high computational complexity of query evaluation in constraint database systems [RSV01] and the

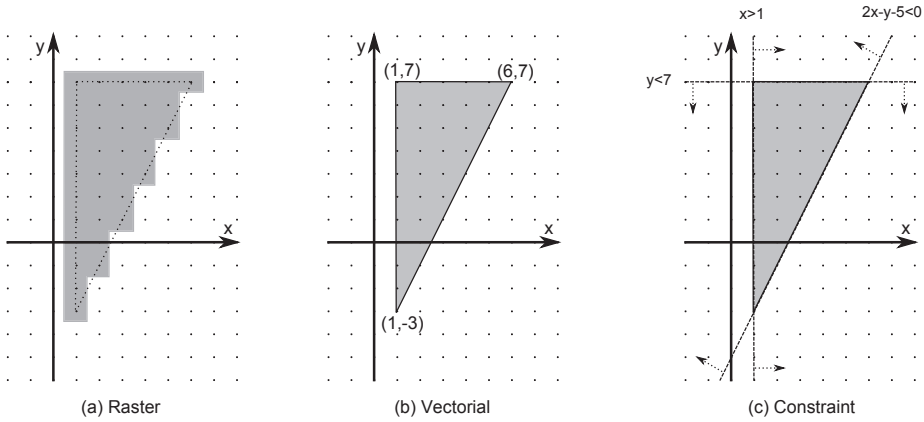

Figure 2.2: Object represented using (a) raster, (b) vectorial or (c) constraint databases models.

difficulty of implementing them in non-constraint database systems (it would require to incorporate a constraints engine just for supporting spatial data).

Figure 2.2 shows the same object represented using a raster, a vectorial and a constraint databases model.

Although all those types of spatial data models are used (each of them in their specific niche of application domains), only two of them are used in general purpose spatial databases and tools: the raster and vectorial models. Raster models are mainly used in imagery processing and domains where the spatial information of interest corresponds with *coverages*. They are also covered by several ISO standards (e.g they are considered in some of the interfaces for coverages defined by the ISO 19123:2005 [ISO05b]). Vectorial models are more widely used on GIS applications where the spatial data to be managed correspond mainly with *geographic objects*. They are also used in some of the international standards (e.g., [ISO07]).

This research work focuses its attention in vectorial data models, given that they are the ones where inconsistency problems are more relevant. Nevertheless, and to put in context the analysis here performed, we compare in Chapter 3 the consistency properties of the different solutions used in vectorial models with the ones exhibited by the other data model types.

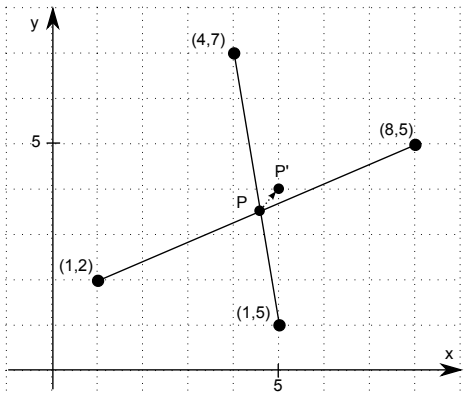

Figure 2.3: Two segments with end-points having integer coordinates. Their intersection point has non-integer coordinates.

2.4 Physical spatial models

The main problem in defining physical spatial models that translate the expressiveness of abstract and discrete spatial models to computer applications (that is, commercial spatial database technologies) is that abstract and discrete spatial models are usually defined over a continuous domain, but for their implementation we need to define a physical spatial model that uses discrete representations for the spatial types. For example, a value of type *point* [ISO03] can be represented at physical level as a pair of coordinates, each of them represented as a 32 bit signed integer number. A *curve* can be represented as a sequence of *point* values defining a polyline. And *surface* can be represented as a polygon, which in turn is represented as the sequence of *point* values representing its boundary.

The problems arising from the need to use a discrete space at physical level are far from trivial. For example, if we decide to use the previously described discrete space (coordinates represented as 32 bit signed integer numbers), then the representation space is no longer closed under the set-theory operations (e.g., union or intersection). This happens because the intersection point of two segments defined between points represented with integer coordinates does not usually have integer coordinates (an example is shown in Figure 2.3). The straightforward solution to such problems (as for example, to approximate the non-representable points by the closest representable

ones) makes the operations closed,[1] but it violates the basic properties of set-theory. For example, for regions A and B, the following relationships $A \subseteq (A \cup B)$, $(A \cap B) \subseteq A$ or $(A \setminus B) \cap B = \emptyset$ do not hold any more, generating inconsistencies between the answers. In certain domains (e.g., graphical user interfaces) such rounding errors may be acceptable, but they cannot be tolerated in query evaluation for spatial analysis, since they may lead to wrong answers. For example, the intersection of a river and a highway may be found to lie neither on the river nor on the highway.

Depending on their approach to this problem, spatial extensions for commercial databases can be classified in one of the following two groups:

1. They do not provide operations that are not closed under the selected discrete representation. This usually means that they provide only predicates over spatial data types and operations for composing and decomposing such objects (e.g., operations for constructing the segment between two given points).

2. They provide the whole set of operations, but for those operations that are not closed over the selected discrete representation space they return an approximation of the result.

Examples of the first group of spatial implementations were the first generations of spatial databases extensions, such as *Illustra 2D Spatial Datablade* [Ill94], earlier versions of Oracle's spatial extension (*Oracle8 Spatial Cartridge*) [Ora97] and MySQL 5.5 [Mys10]. *Illustra's* extension [Ill94] provided data types *Point*, *Line* (in the mathematical sense), *Segment*, *Path* (a polyline), *Polygon* and *Polygon Set* (a region as a set of polygons with holes), apart from some other data types such as *Circle*, *Ellipse* or *Box*. The provided operations were basically predicates and operations for decomposing/constructing a value (for example, retrieving the *n*-th *Segment* of a *Path* or constructing the *Segment* having as end points two given *Point* values). Any operations that became non-closed over the discrete representation used, such as the set-theory operations, were not provided.[2] Current MySQL 5.5 provides also spatial extensions [Mys10]. However, it only provides spatial operations (returning spatial values) that are already expected to return approximated results (e.g., *Centroid()*, *PointOnSurface()*, etc.), decomposition operations (retrieving elements of the representation of other object, e.g., *Boundary()*, *StartPoint()*, *EndPoint()*, *PointN()*, *ExteriorRing()*, *InteriorRingN()*, etc.) and constructor operations (building another

[1] Artificially closed, because now we are not implementing the *intersection* operation (for instance), but an *approximate intersection* operation.

[2] The only exception to this are operations that are already expected to return approximate values, as getting the approximation of an *Ellipse* value as a *Polygon* or getting the bounding box of any spatial value.

spatial object representation from existing ones, e.g., *Point()*, *LineString()*, *Polygon()*, etc.).

Examples belonging to the second group are more modern generations of spatial databases, such as *DB2's Spatial Extender* [Dav98] (by *ESRI*, available also under *Informix*), more modern versions of Oracle's extension (*Oracle8i Spatial Cartridge*) [Ora99, Ora10] and most of current spatial databases (Oracle Spatial 11g [Ora10], PostgreSql/PostGIS [Ref10] and SQL Server 2008 [Mic09]).

The problem of robustness and topological correctness of geometric computation has also been addressed in the computational geometry literature [DS90, For85]. The literature distinguishes between perturbation-free approaches where the idea is to perform geometric computations with sufficiently high precision so that no errors occur (e.g., [KM83, OTU87]) and perturbation (approximation) approaches (e.g., [DS90, GM95, Mil89, Sch94]) that allow to slightly change the input data of computations (in order to reduce errors in the computations) or the results (in order to be able to represent data at a fixed level of precision).

A special application of the perturbation approach within the area of spatial databases is used in the ROSE algebra [GS95]. The ROSE algebra uses an underlying discrete geometric basis called a *realm* [GS93]. Intuitively, a realm contains a consistent representation of all geometric data of an application. All numerical problems are treated at the realm level, which uses a particular perturbation method. Values of spatial data types are defined on top of realms which ensures that all operations (including the set-theory operations) return consistent answers (for example, all the relationships $A \subseteq (A \cup B)$, $(A \cap B) \subseteq A$ or $(A \setminus B) \cap B = \emptyset$ hold in the implementation).

In Section 2.4.1 we show three representative examples on how problems arising form space discretization are addressed in commercial and research solutions. A more detailed explanation on how *realms* work is shown in Section 2.4.2.

2.4.1 Commercial approaches

Among current commercial solutions, *PostGIS*, *Oracle Spatial* and *Microsoft SQL Server* are the three more representative and used. PostGIS is the most commonly used free (GPL) spatial database extension (used with PostgreSQL), whereas Oracle spatial is a reference among commercial (proprietary) spatial database extension (used with Oracle DBMS). The spatial support is a relatively new feature in *Microsoft SQL Server*, although already fulfilling most spatial standards. All of them follow (in one way or another) the perturbation/approximation approach and are representative of the current state of the art in current commercial spatial database technologies. And all of them fail to provide appropriate consistency among spatial operations. Appendix A shows a

very simple example that causes all of them to return answers that are inconsistent with the properties expected from their abstract spatial models.

2.4.1.1 PostGIS

PostgreSQL (http://www.postgresql.org/) is one of the more widely used free (non commercial) database management systems. It started as an evolution of the original Postgres project (directed by Michael Stonebraker), evolving from it to a fully SQL92 compliant DBMS. Postgres was the first prototype of extensible DBMS, allowing users to extend it with new modules of data types and functions.

PostGIS (`http://postgis.refractions.net/`) is a free (GPL) spatial extension for PostgreSQL developed by Refractions Research. It extends PostgreSQL with new spatial data types and functions fully implementing the OGC SFS standard.

PostGIS uses a discrete coordinates space, using double-precision floating-point numbers for representing the coordinates [Ref10]. Initially it restricted operations, avoiding the implementation of the operations that became non closed with the space discretization. However, later versions incorporated them using the GEOS engine (`http://geos.refractions.net/`), a port to C++ of the JTS Topology Suite [Viv03] (`http://www.vividsolutions.com/jts/`). GEOS (as JTS) is a library designed with high care in numerical stability, trying to minimize errors arising from the space discretization. PostGIS/GEOS uses the perturbation approach, approximating the results of operations. It also uses some techniques to increase its reliability and improve numerical stability in computations [Viv03]. For example:

- It implements algorithms for boolean operators that ensure that the answer (true/false) is correct, using an implementation of [ABD+97].
- It uses *translation* techniques to improve the probabilities of correct generation of result geometries. When a spatial operation that should return a geometry generates a *Topological Exception*, it performs the operation with a translation of the problem (input arguments) to the origin of the axis, therefore removing the common bits between the input coordinates. This translation results in a reduction of the significant bits of the coordinates, leaving extra precision bits available to reduce rounding errors in computations. This reduces the likelihood that precision errors lead the algorithm to unexpected states, making it more likely that the result can be computed this second time. This *translated* result is then corrected again to recover the *common bits* removed, and the resulting geometry is then returned.

Even though these techniques improve the reliability of PostGIS, they do not always guarantee the correct behavior, nor do they guarantee consistency among operations.

Moreover, PostGIS does not handle possible topological errors produced by the approximation of results or numerical precision errors. As a result of all this, the combination of operations (a spatial operation having as argument the result of another spatial operation) can lead to *Topological Exception* in GEOS. These errors where not properly handled by PostGIS 1.0, because a single topology exception in a query would abort the whole query without returning any result. Starting in PostGIS 1.5 and later, these errors are handled returning a null geometry in those cases where a *Topological Exception* was generated. At least, this way the query does not get interrupted. A better view of these errors and how often they appear is shown later (in Table 4.1).

2.4.1.2 Oracle Spatial

Oracle Database, commonly known as Oracle RDBMS or simply as Oracle, is an object-relational database management system developed by Oracle Corporation. Its first version was commercialized in 1979. Oracle Spatial [Ora10] is an extension module (licensed as an option to the core Oracle DBMS) that extends Oracle with spatial capabilities. Oracle Spatial was officially launched as an extension to Oracle 7 named "Spatial Data Option" (SDO), and is sold as "Oracle Spatial" since Oracle 8.

Oracle Spatial is conformant with the following International Organization for Standardization (ISO) standards:

- ISO 13249-3 SQL Multimedia and Application Packages - Part 3: Spatial.
- ISO 19101: Geographic information - Reference model (definition of terms and approach).
- ISO 19109: Geographic information - Rules for application schema (called the General Feature Model).
- ISO 19111: Geographic information - Spatial referencing by coordinates (also OGC Abstract specification for coordinate reference systems).
- ISO 19118: Geographic information - Encoding (GML 2.1 and GML 3.1.1).
- ISO 19107: Geographic information - Spatial schema (also OGC Abstract specification for Geometry).

Oracle Spatial provides support for storing, retrieving, querying and analyzing spatial data. It fulfills the OGC Simple Feature Specification SFS [OGC06], providing the whole set of operations on it, including both boolean operators (e.g., intersects or overlap) and set-theory operations (intersection, union, difference, etc.). Oracle Spatial

data can be associated with geodetic (geographical) or cartesian (either projected or local) coordinate systems:

- Geodetic coordinates (sometimes called geographic coordinates) are angular coordinates (longitude and latitude), closely related to spherical polar coordinates, and are defined relative to a particular Earth geodetic datum. (A geodetic datum is a means of representing the figure of the Earth and is the reference for the system of geodetic coordinates).
- Cartesian coordinates are coordinates that measure the position of a point from a defined origin along axes that are perpendicular in the represented two-dimensional or three-dimensional space. If a coordinate system is not explicitly associated with a geometry in Oracle, a Cartesian coordinate system is assumed. Cartesian coordinates can be either projected or local:
  - Projected coordinates are planar Cartesian coordinates that result from performing a mathematical mapping from a point on the Earth's surface to a plane. There are many such mathematical mappings, each used for a particular purpose.
  - Local coordinates are planar Cartesian coordinates in a non-Earth (non-georeferenced) coordinate system. Local coordinate systems are often used for CAD applications and local surveys.

When performing operations on geometries, Oracle Spatial uses either a Cartesian or curvilinear computational model, as appropriate for the coordinate system associated with the spatial data.

Oracle Spatial uses a discrete space for representing spatial objects. Points in the (2D) space are represented as a pair of coordinates (X,Y), where X and Y are fixed-point decimal numbers (Oracle and SQL'92 standard NUMBER data type). Oracle Spatial is an option for Oracle Enterprise Edition, and must be licensed separately. It is not included as an option in the Oracle Standard Edition. Current version is Oracle Spatial 11.2 (for Oracle Database 11g Enterprise Edition).

To include basic support of spatial data in Oracle Database, a limited version of Oracle Spatial (named *Oracle Locator*) is distributed as part of the basic license of Oracle (both in Oracle Standard Edition and Oracle Enterprise Edition). Oracle Locator provides support for spatial data types as well as the basic operations to interact with them, such as operations for querying spatial data properties (e.g., area or length) and logical relations among them (boolean operators, as intersects or overlaps). However, although Oracle claims that both Oracle Spatial and Oracle Locator are fully OGC SFS compliant (and the Open Geospatial Consortium officially recognizes it), the

licensing facts are that it explicitly excludes from the Oracle Locator license, among others, the set-theory operations (e.g., difference, union, etc.). For more information on functionalities licensed with Oracle Locator see "Appendix B: Oracle Locator" in [Ora10].

To address the problems arising form space discretization, Oracle Spatial makes intensive use of *tolerances*. Tolerances define how numerically close two coordinates must be to be considered the same coordinate. With this approach, users and developers can adjust the behavior of Oracle Spatial to the requirements of the data managed (e.g., with which precision the data were capture or which precision is relevant in that specific GIS application). This provides users with a way to handle the imprecisions in their data, but also helps to hide the consequences of using a non-closed discrete space.

Oracle Spatial deals with space discretization problems using a perturbation approach. Tolerances provide both a way of implementing *input perturbations* (assuming that two input coordinates are the same if they are close enough) and *output perturbations* (defining which precision is acceptable in the output). The fact that for set-theory operations Oracle Spatial suggests the tolerances to be higher than a given value evidences that numerical stability problems in the algorithms implementing them requires that an artificially reduced precision has to be considered in the input values.

Anyway, all these impositions do not free users from getting a computing exception due to an *invalid geometry* even though the input geometries are valid. This makes the user (i.e., the GIS software developer) responsible of adjusting the tolerances appropriately and handling the resolution of computing exceptions that may arise. Moreover, the use of *tolerances* does not solve the inconsistency problems, but instead moves them to another place. With tolerances is easy to get the right answer when querying whether an intersection point belongs to the originating segments. But it can return inconsistent answers as that $P_1 = P_2$, $P_2 = P_3$ but $P_1 \neq P_3$, or that $P \in S_1$, $P \in S_2$ but $P \neq (S_1 \cap S_2)$.

Summing up, the way Oracle faces the space discretization problems differ between Oracle Locator and Oracle Spatial:

- With Oracle Locator, the approach is to discard any operation that does not remain closed under the used discrete space.

- With Oracle Spatial, they provide them by using a perturbation approach. The use of tolerance thresholds becomes a flexible way of providing GIS software developers with a method to adjust the behavior of the input and output perturbation techniques implemented in Oracle Spatial. The limitations that Oracle Spatial imposes in the minimum values that can be used as thresholds for the set-theory operations highlight the difficulties of correctly implementing

these operations, and that the coordinate resolution has to be reduced in order to achieve an acceptable stability of the algorithms.

2.4.1.3 Microsoft SQL Server

SQL Server is Microsoft´s commercial relational DBMS. Starting with SQL Server 2008, it provides also support for spatial data, implementing OGC´s Simple Feature Specification [OGC06].

Version 2008 was the first to provided spatial data types. It provides *geometry* and *geographic* data types, allowing computations in the plane and in the sphere. Unlike Oracle, it does not make use of the *tolerance* artifacts, having a functionality closer to the one provided by PostGIS. The syntax of spatial functions, on the other hand, is quite different, using a syntax closer to object oriented languages than the typical for SQL functions.

Although Version 2008 already represented spatial data through double-precision floating-point coordinates, for spatial operations (union, intersection, difference, etc.) it used a precision of only 27 bits in the input spatial data (a precision of roughly 10cm on georeferenced data). This raised complaints because the errors introduced were too evident even when used for visualization applications. Version 2012 extends the precision used to 48 bits, making spatial computations more precise. That precision improvement addresses the complaints regarding the visualization of spatial results, but the inconsistencies among operations remain. Section A.3 shows that, just like in PostGIS and Oracle, even the simplest spatial tests can retrieve inconsistent results, with answers that do not follow the expected spatial logic. Moreover, SQL Server exhibits a rather weird behavior: a tendency to slightly move coordinates in the result, producing the alteration of their less significant bits. This behavior is exhibited even in cases where the coordinate in the results is not a computed coordinate, but instead comes directly from the input arguments, and even if the coordinate value is exactly representable with the 48 bits precision used in the spatial operations. As a result, inconsistencies among operations become more usual than in other spatial database solutions (such as PostGIS and Oracle).

2.4.2 The ROSE Algebra: Realms

In [GS95] the ROSE algebra is proposed. It offers the spatial data types *points* (a set of disconnected points in the plane), *lines* (a set of line segments) and *regions* (a set of non overlapping polygons with holes). The operations provided include a wide set of spatial predicates as well as binary operations over spatial types, including the set-theory operations.

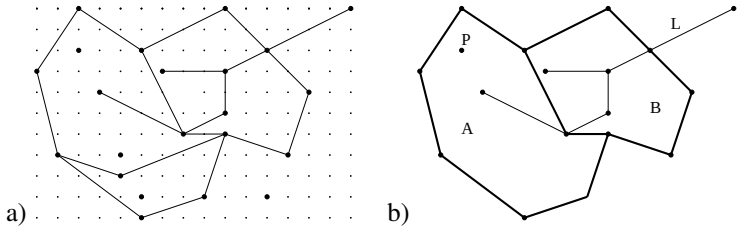

Figure 2.4: Example of objects over a *realm*. a) Elements in a realm. b) Some objects over the realm.

In contrast to other proposals, the definition of the ROSE algebra takes into account the fact that values need to be finitely represented, and introduces an underlying discrete geometric basis called a *realm* for ensuring the consistency of answers returned by the DBMS.

A *realm* is basically a finite set of points and non-intersecting line segments over a discrete domain (a *grid*). Spatial values are represented in the database in terms of points and segments present in the realm. For example, given the realm in Figure 2.4, A and B would be valid *regions* values, L would be a valid *lines* value and P would be a valid *points* value.

Whenever a new spatial object (represented at the lowest level in terms of points and line segments) is inserted into the database, its points and segments are inserted into the *realm* and the spatial object is rewritten in terms of elements of the new *realm*.

For each point or segment that is inserted in the *realm*, some extra steps must be followed to ensure that the resulting *realm* is a valid one:

1. When a new point P is inserted, any segment S_r in the *realm* containing P in its *interior* (i.e., P lies on S_r but it is not an end point of S_r) is split into two new segments connecting the original end points with P. The original segment is replaced by the new ones in the *realm*, and any spatial object using it is modified to reflect this change.

2. When a new segment S is inserted into the *realm*, its end points are also inserted. If S contains a realm point P_r in its interior, it is split at P_r. Furthermore, all intersections of S with segments already present in the realm are determined. For each true intersection point P_i of S with some segment S_i the closest grid point $\overline{P_i}$ is determined. Then, a *redrawing* of S is performed which transforms S into a polyline (a sequence of line segments) passing through all the grid points $\overline{P_i}$. (This process of redrawing is explained in more detail below.) Each segment S_i is

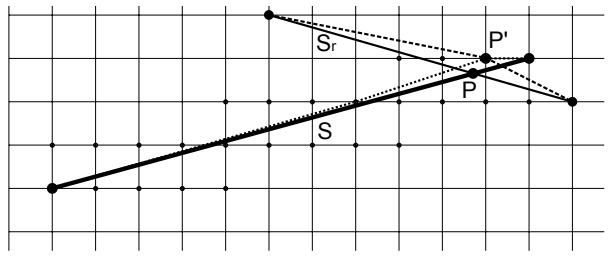

Figure 2.5: Redrawing of a segment S.

redrawn as well to pass through $\overline{P_i}$, its grid intersection point with S. Finally, all changes in segments are propagated to the database to modify the spatial objects using these segments.

To explain the redrawing (originally proposed in [GY86]), the concept of an *envelope* is needed. For a segment S, its envelope consists (roughly) of the grid points immediately below or above it. Figure 2.5 shows the grid points forming the envelope of S.

Intuitively, the process of redrawing can be easily understood if we view the segment S as a rubber band and the points on the envelope of S as nails on a board. When a second segment S_r is found that intersects it in a point P, we grab the rubber band at point P and pull it around the closest nail P'. As a result, the rubber band will touch one or more nails of the board (of the envelope). The resulting list of segments is the redrawing of S. This is also illustrated in Figure 2.5. This approach guarantees that the polyline into which the segment is decomposed always remains within the envelope of the original segment. For more details, see [GS93].

The use of realms as the basis for the representation of spatial objects has the following advantages:

- *It enforces geometric consistency of spatial objects*. For example, the common part of the boundary of two adjacent regions will belong completely to both regions and be exactly the same in both of them.

- *It guarantees closure properties for the spatial objects*. For example, the intersection of two regions A and B can always be represented, because it will be just a new combination of segments of the *realm* belonging to regions A and B.

- *Simplicity and efficiency of spatial operations*. Given that no intersection points between segments need to be detected, the algorithms for implementing spatial operations are more simple and more efficient.

- *It ensures numerical correctness and robustness of geometric computations*. Those problems arise basically in the computation of intersection points of line segments, which in general will not lie on the grid. As there is no pair of segments in the *realm* that intersects, this problem does not arise when performing geometrical computations. All these problems are solved at the realm level, whenever a new value is inserted into the database.

Although the *realms* approach has several advantages, it also has some disadvantages, some of which make the use of a *realm*-based representation in commercial databases (e.g., offering the ROSE algebra as an extension module) difficult:

- *The relationships between points and segments can change after rewriting*. The rewriting of segments can cause points of the *realm* that previously did not lie on the segment to now lie on it. This can change topological relationships between point objects and objects of type *lines* and *regions*. This problem could be solved (as pointed out in [GS93]) by adding the restriction that points of the *realm* cannot lie on the envelope of segments already in the *realm*.

- *Complexity of data structures*. The original implementation proposed in [GS95], with a layer representing the realm and the representation of spatial objects referencing the elements of the *realm* of which they are composed, results in a rather complex representation. The use of *virtual realms*, as proposed in [MPF+96, FDP+99], simplifies it. The spatial objects are directly represented in the database in a *realmized* form (making intersection points explicit), but without further references to *realm* elements. Instead of storing the corresponding *realm* explicitly, the portion of it that is needed is dynamically computed whenever a new insertion is performed, reducing in that way the size of the data stored in the database and the time needed to recover the spatial objects from it.

- *Space overhead*. Due to the rewriting process, a segment of a spatial object can be decomposed into multiple sub-segments in the *realm*. If the overlapping of spatial objects in one area of the plane is high, the overhead produced can lead to a considerable increase in the space needed to store the objects of such an area. Indeed, the fact of representing the object with a higher number of segments will negatively affect the efficiency of the spatial operations. Proposals as the use of *virtual realms* [MPF+96] try to overcome this limitation.

- *Complexity and efficiency of updates*. Given that the update of spatial objects in the database can lead to a cascade of rewritings, the efficiency of update operations can be seriously affected, mainly if the spatial object is inserted into

an area of the plane with a high concentration of objects. Some proposals try to mitigate it by optimizing the order in which segments are used to compute the realm [ZZC+02].

- *Values can change without directly modifying them.* A region A stored in the database can slightly change its value just because the insertion of a second region B has caused the rewriting of some of the segments in the *realm* that compose it. This will be true even if the user that inserted B has no access rights for modifying A.

- *Consistency over time.* Although consistency between answers is guaranteed as long as no new values are introduced in the database, it is not guaranteed between operations performed with different contents of the database. After introducing a new value in the database, the answers of the algebra operations will be consistent again, but perhaps not with respect to the answers given before inserting the new value. For example, the answer to $(A \setminus B) \cup (A \cap B)$ can be different from A if a third object C has been inserted into the database between the computation of $(A \setminus B)$ and $(A \cap B)$. This might complicate transaction management. Moreover, it is a problem for applications in which the time dimension (transaction time) is relevant and hence the consistency of answers over time is also a requirement.

2.5 Analysis and conclusions

In this section the application domain of the spatial databases and geographic information systems (GIS) has been introduced. Current spatial data models (at abstract, discrete and physical level) have been introduced and the basic spatial data types described. It has been shown in detail the problems that arise when trying to translate abstract spatial models to discrete models, physical models and final commercial implementations. Open issues arising from the needs of using a discretized space for coordinates representation have been also outlined. The current state of the art in dealing with the discretization problem has been presented and the strong and week points of those approaches have been shown. Solutions applied in the more representative commercial spatial database extensions (PostGIS and Oracle Spatial) have been described, as well as their limitations. With regard to research proposals, the ROSE-Realms approach has been described in detail, because it is the only one that attempts to "minimize" the key side effects of space discretization, specially with regard to consistence among operations. The reasons that prevent the ROSE-Realms

approach to be adopted by current commercial spatial databases technology have also been identified and analyzed.

The rest of this research work presents two new physical spatial models (Dualgrid and DualgridFF) that address the problems arising from the space discretization (namely, closure among data types and operations and consistency among operations), taking into account at the same time (DualgridFF) the additional requirements of commercial-quality spatial database technologies and standards.

Chapter 3

Consistency of spatial operations

The goal of this chapter is to analyze the existing discrete and physical spatial data models (and their proposed implementations) from the perspective of the consistency among operations[1]. We will show the implications that the selection of the physical spatial model has in the appearance of consistency problems due to breaking the basic properties on which the abstract spatial model relied.

The chapter is structured as follows: Section 3.1 highlights the relevance of consistency problems in the development of spatial applications. Then, Section 3.2 reviews several techniques used in the implementation of vectorial spatial databases to deal with inconsistencies. After that, Section 3.3 compares the consistency properties of the representation models and approaches that rely on the techniques described in Section 3.2. Finally, Section 3.4 concludes the chapter.

3.1 Understanding the relevance of consistency in the development of applications

Abstract spatial data models (introduced in Section 2.2) are usually defined over a continuous space, usually $\mathbb{R}^n$. This helps to define in a formal way intuitive models, focusing in what needs to be represented (data types) and how the users will need to process it (operations). The intuitiveness of the abstract model allows the user to easily

[1] For the sake of clarity, from now on we will call it just *consistency*.

understand and interact with it. For example, if he asks for the intersection point P of two curves $C_1$and C_2, he can safely assume that $P \in C_1$ and $P \in C_2$. And if he asks for the area R of a province P whose land salinity is lower than x, he can be sure that $R \subseteq P$.

Discrete models (described in Section 2.3) take a step towards the implementation of abstract models in computers by defining the representation of data types in terms of finite basic elements. They also take care that the proposed logical representation is appropriate for the efficient implementation of the data model operations. However, they do not enforce the use of a given data structure or algorithm.

Physical models (presented in Section 2.4) finish the process of defining a computer implementable model: they define the exact representation for the data types, including finite representations for the basic elements of the data type defined at the discrete model, and efficient data structures to store and process them. Therefore, defining a spatial physical model implies that the infinite continuous space assumed at abstract and discrete level must be replaced by some kind of finite approximation (a *space discretization*), so that spatial basic elements can be represented using a limited storage space. For example, one can use a space based on a grid of integer coordinates K^n, where $K = \{-n, \ldots, -1, 0, 1, \ldots, n\}$, or a similar one using IEEE754 floating-point numbers. However, as a result of such an approximation, the closure properties of the original (continuous) abstract model are usually broken. For example, in the physical implementation of a discrete vectorial spatial model using a space based on a grid of integer coordinates, the intersection of two segments can yield as result a point whose coordinates do not belong to K^n (see Figure 2.3). Figure 3.1 shows another example where the intersection between polygons R_A and R_B is approximated to the underlying grid (let us call this approximated intersection R_{AB}). We can see that the approximated intersection has the following incoherencies:

- Point P_{1a} is in the boundary of R_{AB}, even though it does not belong to R_A or R_B.
- Points P_{2a}, P_{2b} and P_{2c} should belong to the interior of R_{AB} (because they belong to the interior of both R_A and R_B), but they instead belong to its boundary.
- Points P_{3a}, P_{3b}, P_{3c} and P_{3d} should belong to the boundary of R_{AB} (because they belong to the interior of one of the regions and the boundary of the other), but they do not belong to R_{AB} at all.
- Points P_{4a}, P_{4b} and P_{4c} do not belong to R_{AB}, even though they belong to the interior of both R_A and R_B.

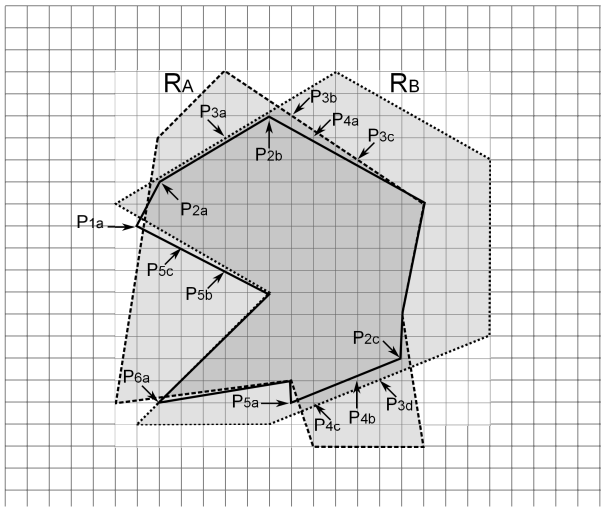

Figure 3.1: Example of errors in intersection operations due to space discretization and approximation.

- Points P_{5a}, P_{5b} and P_{5c} belong to the boundary of R_{AB}, but they belong only to the interior of one of the regions and therefore they should not belong to the intersection.

- Point P_{6a} belongs to the boundary of R_{AB}, but it belongs only to the boundary of one of the regions and therefore it should not belong to the intersection.

The significance of the problems arising from such a *space discretization* depends on the discrete data model chosen and the relevance that the inconsistencies generated by the physical model have on the target application domain.

For applications using raster models consistency among operations has usually low relevance because these applications tend to require techniques closer to *image processing* than to *spatial reasoning*. Also, set-theory operations are always consistent under this model because all the points of the discrete space are always explicitly represented, and no new space points are generated as a result of any operation.

Constraint database models have few consistency problems. They allow the consistent implementation of set-theory operations because the set of constraints resulting from an operation is composed by a subset of the constraints of the input

arguments. Consistency problems arise only when exact point coordinates need to be computed.

Vectorial discrete models are widely used in GIS applications. Their main advantage over raster and constraint models is that there are efficient algorithms in the literature for implementing vectorial operations, and that they are more suitable to express additional information, as for example topological relationships. However, one of the main problems of vectorial models is that, unlike raster and constraint database models, they are very sensible to consistency problems due to space discretizations. This makes it difficult to implement spatial analysis and reasoning applications over them. Therefore, they are the family of discrete spatial models with a wider literature and proposals to deal with (and to mitigate) consistency problem. We analyze the more relevant proposals in the following section.

3.2 Approaches to deal with inconsistency in vectorial spatial data models

As explained previously, the process of defining discrete and physical models that implement abstract spatial data models implies the definition of a discrete space over which spatial objects are represented. The consequences of such a *space discretization* become specially severe for implementations of vectorial discrete models, where the space discretization breaks the properties of the space on which the closure of the abstract and vectorial discrete models rely. As an example, we have shown in Figure 2.3 that the intersection of two segments cannot always be represented over the selected discrete space.

There are two simple and (at first glance) straightforward approaches to deal with this problem: either to provide only those operations that remain closed in the new (discrete) space, or to provide *approximated operations* that return a representable result acceptably close to the theoretical one. The first solution has the disadvantage of severely restricting the power of the original data model, therefore making it less useful. The second solution has several consistency drawbacks, as we will show on the following paragraphs.

First of all, the use of approximated operations implies that they will be inconsistent with the properties that they are supposed to fulfill. For example, it can happen that the intersection point P' of two segments does not belong to any of them (Figure 2.3), or that the overlapping area of two polygons is not completely contained within them (Figure 3.1). Even this *small detail* breaking the model properties can have disastrous consequences. The reason is that spatial algorithms rely on that *small*

detail, and therefore the implementation of spatial operations will not consist in a direct and (more or less) straightforward translation of theoretical geometry algorithms to a programming language. If we do so, such a direct translation would easily reach unexpected states and fail. Instead, their algorithms need to be redesigned to ensure that, regardless of these inconsistencies, they will be robust (always finish) and return topologically correct results. This adds a higher degree of complexity to the implementation of spatial database technologies.

Moreover, even if such goals are achieved, and we manage to properly implement those operations, the inconsistencies among operation results are unavoidable. For example, as shown in Figure 3.1, due to those approximations the result of $R_A \cap R_B$ could contain points that do not belong to R_A, points that do not belong to R_B or even points that do not belong to any of them (see point P_1 in the figure). Furthermore, it could even fail to contain points that belong to both of them (see point P_2 in the figure).

Whether any of the two solutions described above is acceptable depends mainly on the application domain where the resulting implementation is used.

If the spatial data model is basically used to display the spatial objects, (e.g., to represent in a map the location of one facility or the geographic extension of one parcel), then inconsistencies are in general perfectly acceptable and, for example, it does not matter if the intersection of two regions is only an approximation, as far as it is accurate enough to be indistinguishable by the user from the theoretical one. In those domains, it is usually more interesting to provide a powerful set of operations than to ensure the consistency among the operations provided.

However, for target application domains where the provided operations are going to be used to build more complex algorithms, then the use of approximated operations can be a big handicap. As in the case of the implementation of the approximated spatial operations, the new algorithms will have to be carefully designed to deal with the inconsistencies returned by the data model operations, increasing the complexity of the final application. The disadvantage is specially relevant in those applications where some kind of automated reasoning over the spatial data is performed. On these domains, it is usually preferable to provide a smaller set of operations, less powerful but with a consistent behavior. However, such a set of operations must be at the same time large enough to provide the minimal functionalities required by the target application domain.

The set of operations that can be provided together consistently depends on the specific discrete space and the spatial representation used. Therefore, these two become a key aspect when selecting the spatial data model implementation to be used for one given application domain. In the following sections, we present the different approaches that can be used to define a consistent physical data model.

3.2.1 Restriction of operations

The idea of this approach is that the algebra will provide only a set of operations that remain closed over the physical implementation. With that purpose, a subset of operations of the original algebra is removed, to ensure that the remaining set is closed. The specific set of operations to be removed depends on both the properties of the data model and the relevance of the different operations for the target application domain.

One disadvantage of models following this approach is that they drastically reduce the set of available operations. For example, they usually do not provide set-theory operations. The operations that they provide are usually restricted to spatial predicates (e.g., topology relationships), decomposition and construction operations (pointN(*line*), boundary(*region*), line(*point[]*), etc.) and numerical operations (e.g., distance, area, etc.). Current examples of this approach are MySQL (since version 5.0 provides spatial support) and Oracle Locator extensions to Oracle DBMS (due to its licensing policy).

3.2.2 Restriction to orthogonal boundaries

If the only spatial objects allowed are those with boundaries composed of segments orthogonal to the plane axes, then the model remains closed under most of the spatial operations. For example, set-theory operations are closed.

An example of this approach is [LTR99]. The advantages of this approach are its simplicity (both the approach itself and the implementation of most of the operations over it) and that the implementation of algorithms does not even have to address numerical problems.. The disadvantages are that it reduces the expressiveness of the spatial representation, and implies a high storage and computational overhead for representing complex objects with an acceptable accuracy.

3.2.3 Approximated operations

As we cannot return the exact result of an operation because it is not representable (e.g., a point of the boundary in a intersection of lines is not a point of the discrete space), we return a very close approximation (e.g., representing it as the discrete space point closest to that intersection point). If we view this approach as a *result approximation*, we tend to think that the consequences are irrelevant because whether a point is 10 nanometers to the left or to the right is usually of no concern (we wouldn't be able to notice it anyway). However, if we see it as an *approximated operation*, the perspective changes. Now, the *intersection* operation becomes an *approximated intersection operation* whose result could contain some points that only belong to one

of (or even none of) the input objects, and could even not contain some points belonging to both arguments. If we describe it in such a way (that is in fact equivalent to the *result approximation* description), it clearly shows the risks of performing any spatial reasoning using such operations.

As a result of this, *approximated operations* are acceptable in application domains where the properties of the operations are not that relevant (e.g., spatial information visualization). However, there are some other domains (e.g., automated spatial reasoning, spatial knowledge discovery, etc.) where *approximated operations* are not a suitable solution, as such applications rely on the theoretical properties of these operations. Almost all current commercial solutions follow this approach (except a few that follow the restriction of operations approach). The three commercial solutions described in Section 2.4.1 follow the approximated operations approach.

3.2.4 Exact representation

In this approach, the continuous space $\mathbb{R}^n$ is replaced by a $\mathbb{Q}^n$ space that uses variable length rationals are used to represent spatial coordinates. The advantage of this approach is that (almost) all the geometric operations that are closed in $\mathbb{R}^n$ are also closed in $\mathbb{Q}^n$, and therefore the source of inconsistency problems in the implementation of spatial models disappears. The coordinates of such a space are still (in theory) numbers of infinite-size. However, considering that the original spatial data are represented using coordinates of fixed-size, the result of operations will be represented with coordinates of finite-size (proportional to the size of the original coordinates and the number of operations applied to them). Therefore, it will be possible to represent them in a computer. An example of this approach is [WS05].

The disadvantages of this approach are the storage overhead due to the use of larger and variable-size coordinate values, and the computational overhead due to not using the integer or floating-point numbers supported by the arithmetic unit of the computer[2]. Also, the complexity of the required data structures increases: compact representations (records) are more efficient and usually preferred in database implementations, but the use of variable-size coordinates makes it necessary to use more complex data structures. Therefore, this approach is sometimes used in geometric computation, but not in database management systems.

[2]Due to the use of large numbers. The appropriated use of numeric filters can reduce it.

3.2.5 Realms approach

In this approach, described in Section 2.4.2 and proposed in [GS93, GS95], all intersections among spatial objects stored in the database are made explicit at insertion/update time, slightly modifying the spatial objects if needed. As a result, there will be no intersecting segments in the internal representation.

The advantage of this approach is that consistency among operations over static sets of data is guaranteed for a wide set of spatial operations, including set-theory operations. Moreover, fixed-size representation for coordinates (e.g., integers) can be used, and efficient algorithms have been already defined for it [GdRS95, MPF+96]. As a disadvantage, consistency is not guaranteed when the set of data stored in the database changes (e.g., updates). Also, the implementation of object access permissions and transaction management can become tricky in databases using this approach, given that one spatial object A can be modified due to the insertion of another object B. A deeper explanation of this approach, its advantages and disadvantages, has already been given in Section 2.4.2.

An example of algebra implemented using realms is the ROSE algebra [GS95].

3.3 Comparison of consistency support in the spatial dimension

In order to compare the power of the different approaches to address the consistency problems, it is interesting to analyze their capability to provide consistency over the different sets of operations that are typical in the different application domains. That way, given an application domain and the set of operations over which consistency is relevant in that domain, it would be possible to analyze which data model families and/or consistency solutions (in the case of the vectorial model) are the more suitable ones for that domain.

With that purpose, in Subsection 3.3.1 we will first classify the most common spatial operations according to they properties and then, in Subsection 3.3.2, we will analyze the capability of the different spatial models and approaches to provide consistent answers for each family of operations. A more comprehensive analysis can be found in [Cot01].

3.3.1 Operations classification

For this analysis we will classify the spatial operations usually provided in spatial algebras in the following classes[3]:

- *Spatial predicates*. Operations in this family return a boolean value depending on whether their arguments have some spatial relationship. Examples of these operations are *equals*, *disjoint*, *touches*, *within*, *overlaps*, *crosses*, *intersects*, *contains* and *relate.*
- *Set-theory operations*. Operations implementing the set-theory operators over spatial data, like *intersection*, *union*, *difference*. and *symDifference*.
- *Decomposition operations*. These operations decompose spatial objects in the components of their boundary. Examples of these operations are *boundary*, *x*, *y*, *pointN*, *exteriorRing*, *interiorRingN* and *geometryN*.
- *Construction operations*. They build new spatial objects, usually from their components. Examples are *envelope*, *centroid*, *pointOnSurface*, *buffer* and *convexHull*.
- *Numeric operations*. They return numeric results. Examples are *length*, *area* and *distance*.
- *Scaling and rotation operations*.
- *Translation operations*.

3.3.2 Consistency analysis

Table 3.1 shows the consistency properties of the different data models for each set of operations:

- *C* indicates that the set of operation can be provided in the data model with consistency.
- *NC* (Not Consistent) indicates that the operations can be provided in the data model, but without consistency.
- − indicates that the operation cannot be provided and/or has no sense in that model.

[3]For the sake of clarity, names of some operations belonging to each group are given, using (when possible) the names proposed in the OGC-SFS standard [OGC06].

	Raster models	Constraint models	Vectorial models				
			Restric. of operations	Restric. to orthogonal	Approx.	Arbitrary Prec. Rationals	Realms
Spatial predicates	C	C	C	C	C	C	C
Set-theory operations	C	C	-	C	NC	C	C
Decomposition operations	-	C/NC(1)	C	C	C	C	C
Construction operations	-	C/NC(2)	C	C/NC(2)	C	C	C/NC(2)
Numeric operations	C	C	C	C	C	C	C
Scaling and rotation operations	NC	NC	-	NC	NC	C/NC(3)	NC
Translation operations	C	NC	C/NC(4)	C/NC(4)	C/NC(4)	C	NC

Table 3.1: Consistency properties of operations for each data model in the spatial domain.

Wherever several levels of consistency are indicated, they have the following meanings:

- (1) Operations x and y can only be provided through an approximated implementation.
- (2) Only a consistent implementation of *envelope* and *pointOnSurface* can be provided.
- (3) Scaling can be provided consistently, but rotations cannot.
- (4) Consistency can be achieved only if a uniform grid is used.

Spatial predicates and *Numeric operations* can be consistently implemented in every family of spatial data models because they do not need to return spatial objects.

Numeric operations can also be consistently provided in all of them, as they do not return spatial objects either.

Set-theory operations can be consistently implemented in raster and constraint database models. In the vectorial model, the non-consistent versions of the operations

can be implemented in all of the approaches except in the approach of restricting the operations, because its philosophy does not allow approximate results. However, only the restriction to orthogonal, exact computation and Realms based approaches can provide consistency among these operations. The restriction to orthogonal approach ensures consistency because orthogonality implies that X and Y coordinates of any computed point will correspond with X and Y coordinates of some point of the input argument (and hence is guaranteed to be representable). The exact computation approach uses the precision required to exactly represent the result coordinates. Finally, in the Realms approach boundary intersection points have already been made explicit at insertion time, so no new points need to be computed when performing set-theory operations and their result is guaranteed to be exactly representable.

Decomposition operations have usually no meaning in the raster model, where a region is just a collection of raster points and has no boundary. They can be implemented consistently in the vectorial model because spatial objects are already stored representing their boundary, which is usually the basis of the decomposition operations. In the constraint database model they can also be consistently implemented, except for the x and y operations that can only be provided as an approximated operation.

Construction operations have again no meaning in raster models because no topology is explicitly identified in them. For vectorial models, they can be consistently implemented in the approximation approach because a segment can be defined without restrictions between any two representable points. The restriction of operations approach can also implement them consistently. Finally, constraint models, the restriction to orthogonal boundaries approach and Realms can provide a consistent implementation of *envelope* and *pointOnSurface*, but only an approximated one for *centroid*, *buffer* and *convexHull*[4].

Scaling and rotation operations can be provided by constraint databases, vectorial and raster models, although none of them in their consistent version.

Finally, *translation operations* can be provided (in their consistent version) in raster models, whereas constraint database models can only provide them with approximated implementations. Vectorial models can also provide them, although the type of grid used for the representation determines whether consistent or approximated operations must be implemented. If the grid is uniform (always the same distance between adjacent points, as for example when using integer coordinates), then consistency can be provided. In a similar way, and due to the rewriting process of making

[4]However, in the Realms based approach either such an approximated version of *convexHull* takes care that the resulting region remains convex after inserting it in the underlying *realm*, or the possibility of it returning a non-convex *convexHull* should be considered acceptable. Otherwise, one should consider that Realms would not be able to implement *convexHull*.

intersection points explicit, the Realms approach can only provide an approximated implementation.

Of course, the arbitrary-size rationals approach can provide almost all of these operations consistently. The only exception are rotation operations, given that they can generate non-rational coordinates.

It can be seen that, except for the arbitrary-size rationals approach, the models that provide a better support for the consistent implementation of spatial algebras are the constraint model and the vectorial model following the Realms based approach. They provide the wider range of consistency for the most usual spatial operations. However, their weak points are the construction and translation operations. Moreover, we must not forget that Realms ensure such a consistency as far as no new objects are inserted in the database between operation and operation. Other approaches can provide also a wide set of operations, but their capabilities to provide consistency are more limited.

3.4 Conclusions

We have shown in this chapter the differences among the most common spatial representation models with regard to their capabilities to handle consistency among operations. In the case of the vectorial model (the most prone to experience consistency problems), we have analyzed the capabilities provided by the more common/promising approaches followed in their implementation.

It has been shown that the representation model and approach used determines the sets of operations over which consistency can be ensured. Therefore, when implementing a physical spatial model focused on a specific set of application domains, it is worthwhile to analyze the consistency requirements of those domains and take them into account to choose the more appropriate representation model.

The analysis in Section 3.3 shows that the capabilities of several implementation approaches for vectorial models are close in consistency terms to constraint database models. However, when the consistency of set-theory operations becomes important, none of the existing vectorial approaches provides a fully acceptable solution:

- The *arbitrary precision* approach offers consistency together with a powerful representation, but at a high computation cost and difficulties to be adopted in commercial spatial databases[5].
- *Realms* provide consistency together with a powerful representation. However consistency is only guaranteed for static datasets, and the complexity of the

[5]Fixed-size data structures are more efficient, simple and usually preferred in databases, but they cannot be directly used with arbitrary precision numbers.

realms machinery makes it difficult to be implemented in commercial spatial databases.

- The *restriction to orthogonal boundaries* approach provides consistency and allows for the use of simpler algorithms, but at a high cost in space and performance for non-simple spatial objects.
- The *approximated operations* approach provides good performance and low storage requirements, but it is not able to provide consistency and makes algorithm implementations much more complex (due to having to deal with all the inconsistencies it generates).

As a result, we have seen that none of existing vectorial approaches provide consistency at a low storage and performance cost and low implementation complexity. Therefore, the goal of this thesis is to propose an approach for the implementation of vectorial models providing a level of consistency similar to the arbitrary precision approach, while using fixed-size coordinates and aiming to storage and performance costs closer to those of the approximated operations approach.

Chapter 4

Dualgrid

In the previous chapter, we have analyzed the most common spatial data representation models and their approaches for dealing with inconsistencies among operations. We have shown that among the vectorial models, the *Realms* approach [GS93] was specially successful in dealing with avoiding any consistency problem among operations. However, we have also shown in Section 2.4.2 that several drawbacks prevent it from being used as the basis for implementing commercial-quality spatial database management systems. The Realms approach only manages to avoid inconsistencies as far as the collection of spatial objects in the database system is static. This means that it is not able to avoid inconsistencies along the time, because updates on the stored spatial data generate inconsistencies among operations computed before and after the updates. This also means that operations involving non-persistent values (spatial values not stored in the database, but provided as a constant in a query) can also yield inconsistent results.

This chapter introduces Dualgrid, an approach that provides all the advantages provided by Realms without any of its drawbacks. Unlike other approaches, which at the physical data model replace the continuous representation space by a discrete space where most spatial operations lose their closure properties, Dualgrid defines a discrete representation space using fixed-size coordinates that manages to keep the main properties of the original abstract spatial data models. As a result, the typical spatial operations (including set-theory operations) remain closed under Dualgrid, solving the source of the consistency problems of current vectorial spatial data models from the basis.

The chapter is structured as follows. Section 4.1 introduces Dualgrid, defines it formally and mathematically proves its main properties. Section 4.2 further analyzes it

from the perspective of its capabilities to interact with external applications and data. The following sections present the results of experimental adaptations to Dualgrid of an implementation of the ROSE Algebra over Realms (Section 4.3) and an implementation of PosgreSQL/PostGIS with GEOS (Section 4.4). Both sections show how the main problems of ROSE/Realms and PostgreSQL/PostGIS can be solved by using an appropriate dualgrid as their representation space. Section 4.5 shows the links between Dualgrid and the Regular Polytope proposal in [Tho07], and analyzes through it the formal logics that a model over Dualgrid is able to fulfill. Finally, Section 4.6 concludes the chapter.

4.1 Definition of Dualgrid

In Chapter 3 we have shown how different approaches deal with the need of physical spatial data models to define finite representations for coordinates and the consequences of such finite representations to keep the closure properties of the abstract spatial data model. We have described the advantages and disadvantages of each approach, and shown that none of them manage to isolate spatial databases users from such problems. Instead, inconsistencies among spatial operations are usual, making the development of spatial algorithms difficult. For example, the most commonly used approach in commercial spatial databases, operations approximation, makes no real efforts to provide consistency among operations, assuming that users do not need it as far as the result is close enough to the theoretical one. We have shown that such an assumption, although probably valid for visualization applications, is not valid for general purpose GIS applications, where the implementation of any spatial algorithm needs to rely on the theoretical properties of spatial operations.

In Section 2.4.2, the Realms approach [GS93] was introduced and described. It was used as the basis for the development of the ROSE algebra [GS95], and it was designed to isolate the consistency problems at the physical level, simplifying the implementation of the ROSE algebra operations whereas keeping the consistency among them. The reason why the use of *Realms* solves most of the problems when implementing the ROSE algebra is that the set of operations of this algebra has a nice property: given the set of segments and points defining the boundary of the arguments of the operation, the result (if it is a spatial value) can be constructed using only segments and points already present in the input arguments, or sub-segments or points resulting from the intersection of input segments. As the redrawing process performed in the insertion of new segments in the *Realm* decomposes them into sub-segments (see Figure 2.5 and its explanation), making the intersection points explicit, the operations

of the ROSE algebra only need to recombine (in the appropriate way) the elements of the input values to build the result.

However, as points are represented using a finite discrete grid (in the original Realms proposal, 32-bits integer coordinates), in the general case the intersection points do not lie on the underlying grid, and they have to be approximated to the closest representable ones. Therefore, the rewriting process modifies the exact value represented. Furthermore, the insertion of new objects can modify again the value of objects previously stored. This is the main reason of the unsolved problems of the *Realm* approach, such as the inconsistencies between answers over time.

The consistency problems arise when the continuous representation space is replaced by a discrete one that breaks the closure properties of the abstract spatial model. A solution to avoid it in vectorial spatial models consists in finding a discrete and finite space to represent spatial data types that ensures that any intersection point (the result of the intersection of two segments) that can appear as result of the abstract model operations can be exactly represented.

If we can define such a discrete space, the consistency problems would never arise because the physical data model would remain closed under all the spatial operations defined. Using such a discrete representation space would allow implementations based on operations approximation, Realms and other approaches to become closed as well. Hence, the consistency problems on any of them would disappear without the need of doing relevant changes on their algorithms.

With this goal, we define Dualgrid, a discrete representation space based in fixed-size coordinates that allows the definition of physical vectorial spatial data models that keep the closure properties of their operations. As the formal statement of the problem, we need to define a discrete representation space consisting of a grid of points G_p and a grid of segments G_s fulfilling the following three properties:

1. The end points of any segment of G_s are points of G_p.
2. For any two intersecting and non collinear segments[1] $S_1, S_2 \in G_s$, their intersection point $P = S_1 \cap S_2$ belongs to G_p.
3. For any segment $S = (P_a, P_b)$, $S \in G_s$ and any $P \in G_p$, if $P \in S$ and $P \notin \{P_a, P_b\}$, then the sub-segments $S_a = (P_a, P)$ and $S_b = (P, P_b)$ belong to G_s.

We will call such a pair of grids G_p and G_s a *dualgrid*. If valid spatial values are restricted to those whose boundary can be represented using only points and segments

[1] For collinear segments, it is obvious that their intersection, if not empty, will be either a point that is an end point of the original segments (if they meet) or a segment whose end points will be end points of the original segments (if they overlap). Therefore, it belongs to G_P.

belonging to G_p and G_s, then the result of the more typical spatial operations of common spatial vectorial data models (for example, the operations in the ROSE algebra or in the SFS standard [OGC06], including set-theory operations) will be representable, that is, the spatial data model will remain closed when implemented over a *dualgrid*. As non-closure is the root of the inconsistencies among operations in vectorial models, this means that consistency between operations (even over time) will also be guaranteed.

The requirements above mean, first of all, that the domain *Coord* over which the coordinates of points in G_p are defined has to be closed under the arithmetic operations $+$, $-$, $*$, and $/$. This requirement is needed to be able to represent the intersection point of two linear segments. A subset of the real numbers that fulfills such a requirement are the rational numbers ($\mathbb{Q}$). Given the obvious need for a finite representation, the coordinate domain must be restricted to a subset of the rational numbers.

Definition 4.1.1. The domain of point coordinates $Coord(n,m)$ is the set of rational numbers whose numerator and denominator can be represented with integer numbers of n and m bits, respectively[2], that is, $Coord(n,m) = \{x \in \mathbb{Q} \mid x = \frac{num}{den}, num, den \in \mathbb{Z}, |num| < 2^n, 0 < den < 2^m, n,m \in \mathbb{N}\}$.

We will see later on that the use of a $Coord(n,m)$ domain for the coordinates will be enough for defining a closed representation.

Definition 4.1.2. A *grid of points* $G_p(n,m)$ is the set of points that can be represented with coordinates in $Coord(n,m)$, that is, $G_p(n,m) = \{(x,y) \mid x,y \in Coord(n,m)\}$.

Proposition 4.1.1. $G_p(n,m) \subseteq G_p(n+i,m+j), \forall i,j \geq 0$.

This is obvious, as any integer (and here numerator and denominator are integer values) that can be represented with x bits can be represented with $x+i$ bits (with $i > 0$).

Once the set of valid points G_p has been defined, the set of valid segments G_s must be defined in such a way that the intersection of two segments (when the result is a point) belongs to G_p.

Definition 4.1.3. A *grid of segments* $G_s(n,m)$ is the set of segments $S = (P_a, P_b)$, such that $P_a, P_b \in G_p(n,m)$ and their supporting lines (the lines over which they are defined) can be represented with equations $A \cdot x + B \cdot y = C$ having integer coefficients, with $|A|$ and $|B|$ smaller than $\sqrt{2^{m-1}}$ and $|C| < \frac{2^{n-1}}{\sqrt{2^{m-1}}}$.

[2] We are intentionally ignoring the bit required for representing the sign of these numbers. To take it into account would only make these explanations more complex. Just keep in mind that whenever we speak about a numeric value the extra bit required for the sign is implicit.

The bounds on the values of the coefficients A, B and C ensure that the intersection point of any pair of segments $S_1, S_2 \in G_s(n,m)$ belongs to $G_p(n,m)$. It will be demonstrated in the following theorems.

Theorem 4.1.1. The intersection of any two segments of $G_s(n,m)$ (if not empty) is either a segment of $G_s(n,m)$ or a point of $G_p(n,m)$. Formally, $\forall S_1, S_2 \in G_s(n,m)$, $(S_1 \cap S_2 = \perp) \vee (S_1 \cap S_2 \in G_s(n,m)) \vee (S_1 \cap S_2 \in G_p(n,m))$.

Proof. If $S_1 \cap S_2$ is a segment, it must belong to $G_s(n,m)$, because the resulting segment must have the same supporting line as S_1 and S_2 (and hence fulfill the requirements for the coefficients) and each of its end points is an end point of either S_1 or S_2, and therefore belongs to $G_p(n,m)$.

If $S_1 \cap S_2$ is a point, either $S_1 \cap S_2$ is an end point of both S_1 and S_2 (because they meet at their end point) or, given $r : A_r x + B_r y = C_r$ and $s : A_s x + B_s y = C_s$ the supporting lines of S_1 and S_2 respectively,

$$S_1 \cap S_2 = P = \left(\frac{\begin{vmatrix} C_r & B_r \\ C_s & B_s \end{vmatrix}}{\begin{vmatrix} A_r & B_r \\ A_s & B_s \end{vmatrix}}, \frac{\begin{vmatrix} A_r & C_r \\ A_s & C_s \end{vmatrix}}{\begin{vmatrix} A_r & B_r \\ A_s & B_s \end{vmatrix}} \right) = \left(\frac{num_x}{den}, \frac{num_y}{den} \right)$$

Let us denote by $\|A\|$, $\|B\|$ and $\|C\|$ the maximum values that $|A|$, $|B|$ and $|C|$ can have, respectively. Then,

$$|num_x| = |B_s C_r - B_r C_s| \leq 2 \cdot \|B\| \cdot \|C\| < 2 \cdot \sqrt{2^{m-1}} \cdot \frac{2^{n-1}}{\sqrt{2^{m-1}}} = 2^n$$

In the same way,

$$|num_y| = |A_r C_s - A_s C_r| \leq 2 \cdot \|A\| \cdot \|C\| < 2 \cdot \sqrt{2^{m-1}} \cdot \frac{2^{n-1}}{\sqrt{2^{m-1}}} = 2^n$$

$$|den| = |A_r B_s - A_s B_r| \leq 2 \cdot \|A\| \cdot \|B\| < 2 \cdot \sqrt{2^{m-1}} \cdot \sqrt{2^{m-1}} = 2^m$$

Therefore, given that $|num_x| < 2^n$, $|num_y| < 2^n$ and $|den| < 2^m$, it is proved that $S_1 \cap S_2 \in G_p(n,m)$. □

Theorem 4.1.2. $\forall i, j \geq 0,\ j \leq 2 \cdot i :\ G_s(n,m) \subseteq G_s(n+i, m+j)$.

Proof. We have already seen that $G_p(n,m) \subseteq G_p(n+i,m+j)$. Hence, all what we have to prove is that for any segment S belonging to $G_s(n,m)$ the coefficients of its supporting line also fulfill the requirements for $G_s(n+i,m+j)$. For coefficient A, it is clear that if $A < \sqrt{2^{m-1}}$ then also $A < \sqrt{2^{(m+j)-1}}$. In the same way, this is shown for coefficient B. For coefficient C, we can see that if $C < \frac{2^{n-1}}{\sqrt{2^{m-1}}}$ then also $C < \frac{2^{(n+i)-1}}{\sqrt{2^{(m+j)-1}}}$, because

$$\frac{2^{(n+i)-1}}{\sqrt{2^{(m+j)-1}}} = \frac{2^{n-1}}{\sqrt{2^{m-1}}} \cdot \frac{2^i}{\sqrt{2^j}}$$

and

$$j \leq 2i \Rightarrow \sqrt{2^j} \leq 2^i \Rightarrow \frac{2^i}{\sqrt{2^j}} \geq 1$$

Hence, $G_s(n,m) \subseteq G_s(n+i,m+j)$. □

Definition 4.1.4. A *dualgrid* $DG(n,m)$ is a pair of grids $G_p(n,m)$ and $G_s(n,m)$, where $G_p(n,m)$ is a grid of points and $G_s(n,m)$ is a grid of segments. Formally speaking, $DG(n,m) = G_p(n,m) \cup G_s(n,m)$.

The *dualgrid* defined above fulfills the three requirements identified previously, and hence its use as the underlying discrete representation space ensures that all the boundary intersections between spatial objects can be represented. Therefore, the implementation of any vectorial discrete model in a physical spatial model that uses Dualgrid as the discrete representation space ensures that the spatial operations remain closed (including set-theory operations) and that there are no consistency problems. Furthermore, the dualgrid details are completely transparent to the user.

In the case of realm-based representations over Dualgrid, it would also guarantee that the insertion of new elements into a *Realm* does not modify the value of any spatial object in it. As a result, it solves the consistency problems over time, as well as most of the remaining problems of the *realms*-based approach.

Corollary 4.1.1. $\forall i, j \geq 0$, $j \leq 2 \cdot i$, any element of a *dualgrid* $DG(n,m)$ is also an element of a *dualgrid* $DG(n+i,m+j)$.

Proof. Obvious given that $G_p(n,m) \subseteq G_p(n+i,m+j)$ and $G_s(n,m) \subseteq G_s(n+i,m+j)$. □

This last property defines the restrictions that have to be fulfilled to ensure that data provided in a specific *dualgrid* resolution can be imported into a table where a different resolution is used (for example, when importing data from another database). It shows that it is possible to improve the resolution of the *dualgrid* used by a set of spatial data without altering such data at all.

4.2 Data importation and exportation

Users would expect the following properties from a spatial database system implementing Dualgrid with regard to import and export operations:

- Data in the database can be exported using a resolution α for points (including end points of segments) and α' for segment slopes.
- Data using a resolution α for points and a resolution α' for segment slopes can be loaded into the database.
- Data using a resolution β for points and end points of segments, with no restrictions for segment slopes, can be loaded into the database.

Fulfilling the first and second properties ensures that data can be exported and re-imported into the database without loss of information. Fulfilling the third property ensures that data in the usual formats (where no restrictions for segment slopes exist) can be loaded into the database. Such properties are important because they ensure that user applications can effectively interact with the spatial database, retrieve and modify spatial objects and store them back in the database, and also because they ensure that existing data can be imported. Whereas the first and second points are already fulfilled by any *dualgrid*, we need to study some further properties of *dualgrids* to be able to determine which subset suitably fulfills the third one.

Theorem 4.2.1. Let $G_p(n,1)$ be a grid of points with integer coordinates (that is, the only valid value for the denominator is 1). Then, $\forall S = (P_a, P_b), P_a, P_b \in G_p(n,1) : S \in G_s(3n+3, 2n+3)$.

Proof. To make the proof more readable, let $n' = 3n+3$ and $m' = 2n+3$. What we have to prove is that $\forall S = (P_a, P_b), P_a, P_b \in G_p(n,1) : S \in G_s(n', m')$.

It is clear that $G_p(n,1) \subset G_p(n', m')$ and hence $P_a, P_b \in G_p(n', m')$. Therefore, the only thing that needs to be proved is that $\forall P_a, P_b \in G_p(n,1)$, the supporting line $r : A \cdot x + B \cdot y = C$ of segment $S = (P_a, P_b)$ has $|A|$ and $|B|$ smaller than $\sqrt{2^{m'-1}} = \sqrt{2^{(2n+3)-1}} = 2^{n+1}$ and $|C| < \frac{2^{n'-1}}{\sqrt{2^{m'-1}}} = \frac{2^{(3n+3)-1}}{\sqrt{2^{(2n+3)-1}}} = 2^{2n+1}$.

Given $P_a = (X_a, Y_a)$ and $P_b = (X_b, Y_b)$, the coefficients of the supporting line are $A = Y_a - Y_b$, $B = X_b - X_a$ and $C = X_b Y_a - X_a Y_b$. It is easy to see that:

$$|A| = |Y_a - Y_b| < 2^n + 2^n = 2^{n+1}$$

$$|B| = |X_b - X_a| < 2^n + 2^n = 2^{n+1}$$

$$|C| = |X_b Y_a - X_a Y_b| < 2^n \cdot 2^n + 2^n \cdot 2^n = 2^{2n+1} \leq 2^{2n+1}$$

□

Theorem 4.2.2. Let $G_p(n,m)$ be a grid of points. Let $P_a = (\frac{num_{ax}}{den_a}, \frac{num_{ay}}{den_a})$ and $P_b = (\frac{num_{bx}}{den_b}, \frac{num_{by}}{den_b})$ be two points with homogeneous[3] coordinates such that $P_a, P_b \in G_p(n,m)$. Let $S = (P_a, P_b)$. Then, $S \in G_s(3n+m+3, 2n+2m+3)$.[4]

Proof. To make the proof more readable let $n' = 3n+m+3$ and $m' = 2n+2m+3$, that is, we have to prove that $S \in G_s(n', m')$.

In the previous proof for points with integer coordinates we have already seen that the supporting line of S is $r : A \cdot x + B \cdot y = C$, with $A = Y_a - Y_b$, $B = X_b - X_a$ and $C = X_b Y_a - X_a Y_b$. In this case,

$$A = \frac{num_{ay}}{den_a} - \frac{num_{by}}{den_b} = \frac{den_b \cdot num_{ay} - den_a \cdot num_{by}}{den_a \cdot den_b}$$

$$B = \frac{num_{bx}}{den_b} - \frac{num_{ax}}{den_a} = \frac{den_a \cdot num_{bx} - den_b \cdot num_{ax}}{den_a \cdot den_b}$$

$$C = \frac{num_{bx}}{den_b} \cdot \frac{num_{ay}}{den_a} - \frac{num_{ax}}{den_a} \cdot \frac{num_{by}}{den_b} = \frac{num_{bx} \cdot num_{ay} - num_{ax} \cdot num_{by}}{den_a \cdot den_b}$$

Such a line is equivalent to the line $r' : A' \cdot x + B' \cdot y = C'$, where

[3] Two coordinates are called *homogeneous* if they have the same denominator.

[4] Note that we give an upper bound for the values of n and m in the required segment grid $G_s(n,m)$, and do not define the minimum for them. In specific cases (as in the one with integer coordinates shown above) a segment grid with slightly smaller sizes for numerator and denominator could be used.

$$A' = A \cdot den_a \cdot den_b = den_b \cdot num_{ay} - den_a \cdot num_{by}$$

$$B' = B \cdot den_a \cdot den_b = den_a \cdot num_{bx} - den_b \cdot num_{ax}$$

$$C' = C \cdot den_a \cdot den_b = num_{bx} \cdot num_{ay} - num_{ax} \cdot num_{by}$$

Hence,

$$\left|A'\right| = \left|den_b \cdot num_{ay} - den_a \cdot num_{by}\right| < 2^{n+m+1}$$

$$\left|B'\right| = \left|den_a \cdot num_{bx} - den_b \cdot num_{ax}\right| < 2^{n+m+1}$$

$$\left|C'\right| = \left|num_{bx} \cdot num_{ay} - num_{ax} \cdot num_{by}\right| < 2^{2n+1}$$

It is easy to see that:

$$\sqrt{2^{m'-1}} = \sqrt{2^{(2n+2m+3)-1}} = 2^{n+m+1} > \left|A'\right|$$

$$\sqrt{2^{m'-1}} = \sqrt{2^{(2n+2m+3)-1}} = 2^{n+m+1} > \left|B'\right|$$

$$\frac{2^{n'-1}}{\sqrt{2^{m'-1}}} = \frac{2^{(3n+m+3)-1}}{\sqrt{2^{(2n+2m+3)-1}}} = \frac{2^{3n+m+2}}{2^{n+m+1}} = 2^{2n+1} > \left|C'\right|$$

Therefore, $S \in G_s(3n+m+3, 2n+2m+3)$. □

As a result of the previous analysis, we can conclude that if we intend to import external data (which will not have restrictions on segment slopes), then we have two basic possibilities. One is to use a *dualgrid* able to load data where the point coordinates are integer numbers of n bits (this would be the resolution β mentioned above). In this case, (as we have seen in the previous analysis) the best option is to use a *dualgrid* $DG(3n+3, 2n+3)$. The other option is to choose a *dualgrid* able to accept external data whose points are represented with homogeneous coordinates (the

same denominator for both coordinates), where n bits are used for the numerator and m for the denominator (this would be another example of a resolution β). In this case, the optimal *dualgrid* would be $DG(3n+m+3, 2n+2m+3)$. Such a *dualgrid* would also accept data with integer coordinates of $n+\frac{m}{3}$ bits. Note that one special case of data provided in such a homogeneous format is the case when point coordinates are represented as fixed point decimal numbers. If i bits are used for the integer part and m bits are used for the decimal part, then we can represent those points with homogeneous coordinates with $i+m$ bits for the numerator and m bits for the denominator.

As an example, suppose that we want to store data represented with UTM coordinates using the spatial reference system EPSG:23029[5], which uses the geodetic datum ED50 and projected coordinates following the Universal Transverse Mercator coordinate system. This spatial reference system is widely used for representing spatial data in the geographical area of continental Portugal and Galicia (northwestern region of Spain). The area covered by EPSG:23029 has projected coordinates (in meters) between 230.000 and 800.000 in the X axis, and between 4.000.000 and 6.700.000 in the Y axis.

Supposing that we want to store data with a precision of 1cm, and hence we decide to represent coordinates in centimeters instead of meters, that means that coordinates will have integer values smaller than 670.000.000. This means that input spatial data without restrictions on the segment slopes requires 30 bit integers for the coordinates, because $2^{29} = 536.870.912 < 670.000.000$ and $2^{30} = 1.073.741.824 > 670.000.000$.

This would mean that input spatial data (without segment slope restrictions, the β resolution above) require 30 bit integers for coordinates. Applying Theorem 4.2.1, this would require a Dualgrid using (at least) rational coordinates with a $3*30+3 = 93$ bits numerator and a $2*30+3 = 63$ bits denominator.

As another example, let suppose now that we want to store data represented using the spatial reference system EPSG:4326[6], which represents coordinates in degrees over a reference ellipsoid using the geodetic datum WGS84. This spatial reference system is widely used, being the coordinate reference system used by the Global Positioning System (GPS) and for NATO military geodetic surveying.

The area covered by EPSG:4326 is the entire world, using spherical coordinates in degrees, with values ranging from -180 to $+180$ in the X axis, and from -90 to $+90$ in the Y axis.

At the equator, 360 degrees are approx. 40.000 km, that is, 4.000.000.000 cm. This means that each degree covers approx. $\frac{4.000.000.000}{360} = 11.111.112$ cm. Hence, to represent the coordinates in degrees with a precision of 1 cm we need 8 bits for the

[5] http://spatialreference.org/ref/epsg/23029/

[6] http://spatialreference.org/ref/epsg/4326/

integer part ($2^8 = 256 > 180$) and 24 bits for the decimal part ($2^{24} = 16.777.216 > 11.111.112$). As all the coordinates in the representation space use a fixed comma, we can instead see the problem as representing the space in $\frac{1}{2^{24}}$ degree units, and hence we can represent the coordinates without comma. This way, the representation of input spatial data coordinates (without segment slope restrictions, the β resolution above) requires $24+8=32$ bit signed integers[7]. Applying Theorem 4.2.1, this requires a Dualgrid using (at least) rational coordinates with a 100 bits numerator ($3*32+3=99$ bits for the absolute value plus 1 bit for the sign) and a $2*32+3=67$ bits denominator.

4.3 Realms and the ROSE Algebra over Dualgrid

In the original implementation proposed for the ROSE algebra [GdRS95, GS95] all the spatial data types are defined over a *Realm*, in which all the intersection points between the boundaries of the spatial objects are made explicit at insertion time, and their values are approximated, if needed, to the closest representable point. The points of the *Realm* are points with unsigned integer coordinates of a given size n (in bits).

It is important to note that the algorithms implementing operations of the ROSE algebra rely on the fact that all intersection points are explicit. These algorithms are generally simpler and more efficient because of this, but they will crash or yield wrong results if they receive arguments with intersecting segments.

The basic idea for implementing a *realm-less* version of the ROSE algebra (which can also be viewed as using an *implicit Realm)* is to represent data over a *dualgrid* such that all intersection points are representable without loss of precision. In addition, we need to make sure that the algorithms receive only arguments that do not have proper segment intersections. In the following paragraphs we first explain how a Realm can be represented over a *dualgrid*, and in the next step, how the (explicit) Realm can be omitted altogether.

For representing a *Realm* over a *dualgrid* $DG(n,m)$, the domain of valid points of the *Realm* must be $G_p(n,m)$, and the set of valid segments must be $G_s(n,m)$. Formally, a *Realm* over a *dualgrid* $DG(n,m)$ is a subset $R = P \cup S$ of $DG(n,m)$ such that:

1. $P \subseteq G_p(n,m)$, $S \subseteq G_s(n,m)$
2. $\forall p \in P \forall s \in S : \neg(p \,\text{in}\, s)$
3. $\forall s,t \in S, s \neq t : \neg(s \text{ intersects } t) \wedge \neg(s \text{ overlaps } t)$

[7] This means using $32+1$ bits, because we need an extra bit for the sign.

The predicates *in*, *intersects*, and *overlaps* are defined precisely in [GS93]. Informally, they mean that a point does not lie on a segment except possibly on an end point, that two segments do not intersect except in their end points, and that two collinear segments can only touch in an end point.

Now, the operations provided for importing data into the database[8] should check that the input data conform to the *dualgrid* requirements (see properties and notations at start of Section 4.2). If the data to be imported is represented with a resolution of type β (e.g., integer coordinates of size i such that $DG(3i+3, 2i+3) \subseteq DG(n,m)$), then no check needs to be performed. This will be the normal case when importing data from external sources. Otherwise, the import algorithm must check that the type of the input coordinates is compatible with the one used in the *dualgrid* (the resolution α for end points) and the supporting lines of all segments of the input values fulfill the restrictions given for segments of a *dualgrid* (the resolution α' for segment slopes). This case will normally only occur when data have been exported previously from a Dualgrid database of the same resolution.

If these conditions are fulfilled, it is guaranteed that intersection points can be represented as *Realm* points. One can now proceed as before inserting data into a *Realm* (splitting segments at intersection points), but with the following advantages:

- Relationships between points and segments are not altered by the redrawing process.
- Insertion of new data does not modify other data in the database. The new inserted data are also inserted without modifying their values.
- Consistency of answers over time is also guaranteed.

Yet, two problems of the *realm*-based approach remain in this case:

- *Complexity of updates*. Although the cascade of rewritings in the *Realm* is reduced when using Dualgrid (because when a segment in the *Realm* is decomposed, the resulting segments still contain the same set of points, and hence the new segments will not intersect extra points or segments of the *Realm*), the rewriting process can still produce a considerable overhead when the new data are located in areas of the plane with a high density of objects.
- *Space overhead*. Although some decompositions of segments are avoided because no modifications of the values represented in the *Realm* are produced

[8] Although they are not part of the ROSE algebra, they should be provided when implementing a spatial extension for a database.

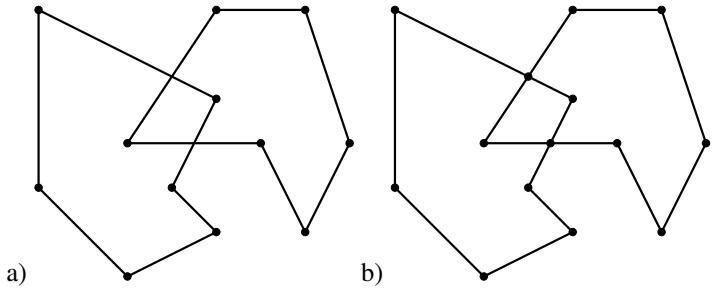

Figure 4.1: Construction of a Realm with the arguments of a ROSE operation. a) Non realm-based arguments. b) Realm-based arguments.

> (and hence there is no risk of triggering an updates cascade), one should still expect a considerable overhead for data located in high density areas of the plane.

However, there exists a better solution: given that now the use of a *Realm* does not modify the data stored in the database, the ROSE operations will return the same value whether the *Realm* is constructed with all the objects of the database or it is constructed only with the arguments of the operation. Hence, instead of storing in the database the *realm*-based representation of the data, we can simply store the data as they are inserted into the database. Whenever a pair of spatial objects is used as argument of an operation, a preprocessing step (*realmizator*) is added that computes a *realm*-based version of the two arguments, where only the intersection points between them are made explicit (any other objects in the database are ignored). This is illustrated in Figure 4.1. This *realm*-based version is then passed to the ROSE algebra algorithm, which computes the result. Such a preprocessing algorithm can, for example, be implemented as a variant of the well-known plane-sweep algorithm by Bentley and Ottmann for finding intersections of line segments [BO79] (see also [PS85]) which requires $O(n \log n + k)$ time where n is the total number of line segments and k the number of intersections. Furthermore, the realmization process only has to be implemented once and it can be applied to all ROSE operations without changing them, instead of having to modify all the ROSE algorithms to detect intersections.

Optionally, a post-processing step (*derealmizator*) can be added to normalize the result (e.g., to melt into one the contiguous segments of the returned object that are collinear). Both pre- and post-processing steps are illustrated in Figure 4.2.

An experimental adaptation to Dualgrid of a ROSE algebra implementation, using the preprocessing algorithm instead of *Realms*, was developed during a research stay of the author of this Thesis at the Fernuniversität Hagen as a proof of concept. Such

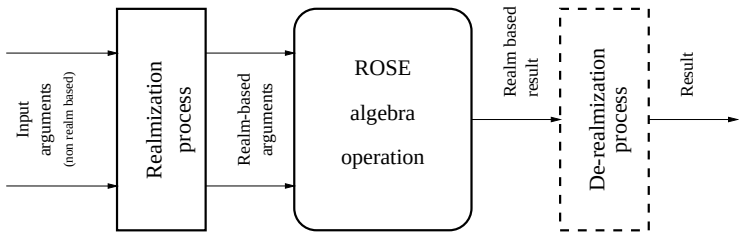

Figure 4.2: Replacement of *Realms* by a preprocessing step before applying an operation.

an implementation was developed as an extension package for *Secondo* and *Informix Illustra*. Both implementations were used for research and teaching purposes at the FernUniversität Hagen. Although many optimization techniques were applied, the implementation was around twenty times slower using Dualgrid than using hardware-supported integer coordinates. The implementation in Secondo was used until a major rewrite of Secondo in 2002.

4.4 PostGIS-GEOS over Dualgrid

In [Cre06] a partial adaptation of PostGIS v1.0.2 to Dualgrid representation was implemented. The implementation focused on the spatial operations and predicates provided by GEOS and the analysis of the impact in robustness and consistency. The resulting implementation was tested against the original PostGIS implementation by performing a series of queries testing the most basic properties of spatial set-theory operations. We used two spatial datasets consisting of two collections of regions delimiting the territory of `comarcas` (Spanish administrative region that groups several adjacent municipalities) and `municipios` (municipalities) on the region of Galicia (Spain). The `comarcas` table had 52 elements, whereas the `municipios` table had 313 elements. Even though the geometries of `municipios` (and respectively `comarcas`) were not supposed to overlap each other, these datasets were created for visualization purposes without any quality checks. This was not a problem for our test because we wanted to test how well PostGIS handles spatial operations between overlapping datasets.

A summary of the test results is shown in Table 4.1. It is important to remark at this point that PostGIS v1.0.2 (on which the Dualgrid-PostGIS implementation was based) produced a *topology exception* for tests 1, 2, 5, 6, 14 and 15 that prevented it from

	Test	% wrong answers (PostGIS v1.0.2)	% wrong answers (Dualgrid-PostGIS)
1	$(A \cap B) - A = \emptyset$	N/A (2,25% in v1.5.11)	0%
2	$(A \cap B) - B = \emptyset$	N/A (2,37% in v1.5.11)	0%
3	$(A \cap B) \subseteq A$	5,66%	0%
4	$(A \cap B) \subseteq B$	5,97%	0%
5	$A \cup (A \cap B) = A$	N/A (2,25% in v1.5.11)	0%
6	$B \cup (A \cap B) = B$	N/A (2,37% in v1.5.11)	0%
7	$(A \cap B) \subseteq (A \cup B)$	0%	0%
8	$A \subseteq (A \cup B)$	2,31%	0%
9	$B \subseteq (A \cup B)$	2,31%	0%
10	$disjoint((A-B),(A \cap B))$	0%	0%
11	$disjoint((B-A),(A \cap B))$	0%	0%
12	$disjoint((A-B),B)$	3,48%	0%
13	$disjoint((B-A),A)$	2,31%	0%
14	$(A \cup B) - B = (A-B)$	N/A (3,48% in v1.5.11)	0%
15	$(A \cup B) - A = (B-A)$	N/A (2,31%in v1.5.11)	0%
16	$(A-B) \subseteq A$	3,08%	0%
17	$B-A \subseteq B$	2,31%	0%
18	$(A \cup B) - (A \cap B) = symdiff(A,B)$	0%	0%
19	$(A-B) \cup (B-A) = symdiff(A,B)$	0%	0%

Table 4.1: Original PostGIS vs Dualgrid PostGIS.

generating a result, whereas PostGIS v1.5.11 handles such cases by returning NULL as the result of the failing operation (we show the test result of PostGIS v1.5.11 on those cases). In all other cases PostGIS v1.5.11 and PostGIS 1.0.2 return the same results.

The test results show that, whereas the original PostGIS implementations failed to behave correctly in almost all the tests (returning wrong results in 2-6% of the answers, depending on the specific test), our modified version of PostGIS using Dualgrid passed all of the tests, retrieving in all cases the right answers. This evidences the capability of Dualgrid to provide consistency to existing implementations, without the need of deep algorithmic changes.

4.5 Rigorous spatial logics over Dualgrid

In [Tho07] the *Regular Polytope* 2D/3D spatial representation is proposed. On it, the segments (2D) and faces (3D) of spatial objects are represented by their line and plane equation coefficients (Ax+By+Cz+D=0), instead of by their vertexes. In the proposed model no boundary points exist. Instead, a spatial object can be directly understood as a point set as in set-theory. A simple decision rule defines whether a point lying in the segments/faces representing the object boundary belongs to the object interior or to its interior.

Apart from defining the Regular Polytope representation, [Tho07] also analyzes the algebraic and logical properties that such a representation can offer, depending on whether an underlying discrete space based on integer, domain restricted rationals (*dr-rationals*) or floating-point numbers is assumed. On such analysis, the concept of dr-rationals follows exactly the approach proposed by Dualgrid, using a *grid1* of integer coordinates points and a *grid2* of finite-size rational coordinates for line/plane intersection coordinates.

In fact, the Regular Polytope in 2D over dr-rationals is completely equivalent to the vectorial representation over Dualgrid with regard to their underlying representation spaces. The three main differences between both are relative to their discrete models:

- The Regular Polytope over dr-rationals stores the line equation coefficients, instead of storing the exact vertex coordinates as Dualgrid does. But as such coordinates can be represented exactly with dr-rationals (equivalent to Dualgrid), and the expressiveness of Regular Polytope is equivalent to that of a vectorial representation, both can represent exactly the same objects.

- The Regular Polytope proposal has *no boundary points*: every point in the space is either in the interior or in the exterior of the spatial object. But in fact, such a property is not related to the Regular Polytope representation or the underlying discrete space used, but instead to a simple interpretation rule that defines (in a logic-consistent fashion) whether a point lying under a line defining the boundary is supposed to belong to the interior or the exterior. Such a rule could also be identically applied over a vectorial representation to get as well a *no boundary points* vectorial model.

- The Regular Polytope proposal allows unbounded geometries. It does so through the straightforward solution of restricting the space limits to the limits of the coordinates representation (from $-M$ to $+M$, being respectively the minimum and maximum value a coordinate can get on their model). A similar approach could be followed in a vectorial representation, and in fact the ISO 19107

standard already allows the definition of unbounded geometries. Dualgrid-based representations could also follow exactly the same approach.

As a result, the same properties apply to Regular Polytopes and vectorial representations over Dualgrid extended with the *no boundary points* additional rule. [Tho07] shows therefore that vectorial representations over Dualgrid (in the *no boundary* interpretation) would support a rigorous algebra that covers the axiomatic definitions of the topological space, metric space, the region connection calculus, the discrete Boolean connection algebra and the weak proximity space.

As an advantage over Dualgrid, the Regular Polytope representation has been defined to support 3D modeling. As a disadvantage, and due to the approach followed by Regular Polytope, the time-complexity of the algorithms proposed for implementing its main operations are, at best, quadratic.

4.6 Conclusions

In this chapter we have presented Dualgrid [CG02], an approach for the representation of spatial objects that solves the robustness problems in the computation of spatial operations and ensures the consistency between answers. This has been done by selecting an appropriate discrete representation space for the spatial values that is completely closed under the set of target operations (e.g., the operations of the ROSE algebra), including set-theory operations. Therefore, it does not impose any change in the discrete spatial model and the algorithms to be used, affecting only to the numerical data types used for the representation of coordinates and for the intermediate computations. We have also shown how the use of a *dualgrid* as the domain set of *Realms* not only solves the open problems of the original implementation of the ROSE algebra, but also simplifies the insertion of elements into the database and allows a very efficient representation of spatial values.

Table 4.2 updates Table 3.1 shown in Chapter 3, showing the consistency properties provided by vectorial models implemented over Dualgrid [Cot01]. Dualgrid provides a consistency level similar to the one provided by *Realms*. Moreover, consistency is provided also over dynamic datasets (not only on static datasets as *Realms* does) and it does not need any of the *Realms* internal data structures. Also, the insertion of a new object has no impact over the other objects representations (object representations are independent), making it easier to implement in transactional databases. Finally, it uses fixed-size coordinates, making it easier to implement in databases than the arbitrary precision rationals approach.

The Dualgrid approach provides closure and consistency between answers for an important set of operations, including the set-theory operations. Although some other

	Raster models	Constraint models	Vectorial models					
			Restric. of operations	Restric. to orthogonal	Approx.	Arbitrary Prec. Rationals	Realms	Dualgrid
Spatial predicates	C	C	C	C	C	C	C	C
Set-theory operations	C	C	-	C	NC	C	C	C
Decomposition operations	-	C/NC(1)	C	C	C	C	C	C
Construction operations	-	C/NC(2)	C	C/NC(2)	C	C	C/NC(2)	C/NC(2)
Numeric operations	C	C	C	C	C	C	C	C
Scaling and rotation operations	NC	NC	-	NC	NC	C/NC(3)	NC	NC
Translation operations	C	NC	C/NC(4)	C/NC(4)	C/NC(4)	C	NC	NC

Table 4.2: Consistency properties of operations for each data model in the spatial domain.

interesting operations are left out[9] (e.g., the *convex-hull* operation or the computation of *Voronoi* diagrams), we expect the consistency between answers for such operations not to be so important as for the set-theory operations. For example, an implementation of the *convex-hull* operation that returns an approximate result but ensures that all the generator points are contained in it is not likely to produce consistency problems between answers, because the more important properties of the result as well as its more important relationships with the arguments are fulfilled.

Finally, let us remark that although the proposal of this chapter is relatively simple and easily implementable, it solves an important problem in offering the only spatial database algebra with guaranteed robustness and correctness properties besides the (original, realm-based) ROSE algebra. Compared to the original ROSE algebra, the relatively complex Realm machinery (especially the redrawing) and its interaction with a DBMS does not need to be implemented any more; yet, the correctness properties of the Dualgrid approach are even better than those of the realm-based ROSE algebra. Our

[9]Such operations are not part of the ROSE algebra because they would not be closed over the Realm basis.

experimental adaptation of the widely used spatial database extension PostGIS-GEOS to use Dualgrid [Cre06], presented in Section 4.4, also shows that it successfully solves all the consistency problems that its implementers were unable to handle. Moreover, Section 4.5 shows how the representation space proposed by Dualgrid has been used as the basis for the development of the Regular Polytope representation in [Tho07]. In it, the Dualgrid representation space is systematically used to support the analysis of the formal logics that Regular Polytope can fulfill, proving the suitability of Dualgrid for providing spatial models with sound formal logic properties.

Dualgrid has its own drawbacks. The most important ones are the performance and storage costs derived from using rational numbers. Moreover, Dualgrid is only suitable for input data that uses integer or fixed-point coordinates. Dualgrid is completely unsuitable for data with floating-point coordinates because of the huge number of bits required to represent the coordinates. In the following chapter we explain this drawbacks and we present an alternative to solve them.

Chapter 5

Dualgrid for floats

In the previous chapter we presented Dualgrid [CG02], a representation approach that defines a physical representation model for spatial databases closed under all types of operations (most importantly, set-theory operations). Although Dualgrid is suitable for domain-specific technologies where interoperability and performance are less relevant than the advantages provided by Dualgrid, it also has some drawbacks that prevent its use in general-purpose spatial database technologies.

In this chapter, we analyze the drawbacks of the original Dualgrid proposal (Section 5.1), and then, we define DualgridFF (Dualgrid For Floats) in Section 5.2. DualgridFF [CL10] goes a step further from Dualgrid by proposing not only a finite discrete representation space, but also a physical spatial representation model that overcomes the drawbacks of Dualgrid, and it allows efficient implementations capable of interacting with other vectorial spatial technologies based in floating-point coordinates. The result is a physical spatial model suitable for its use in commercial general-purpose spatial database technologies. Finally, Section 5.3 estimates the storage and computational costs of using DualgridFF. It also provides useful performance improvement tips and analyzes some interoperability aspects.

5.1 Original Dualgrid drawbacks

Although the original Dualgrid proposal succeeded in providing a finite discrete representation space that kept the closure properties of discrete vectorial spatial models, it had some drawbacks that prevented its adoption in real spatial database management systems. Those drawbacks are related with interoperability and performance, and make

it clear that if we want to translate the advantages of Dualgrid to commercial spatial technologies, we have to a solve these two additional problems.

5.1.1 Interoperability

Current spatial interoperability standards [OGC06], SDBMS [Ref10, Mys10, Ora10, Mic09], technologies [Viv03] and existing spatial datasets are implemented using double-precision floating-point coordinates. The main reasons for floating-point coordinates becoming the de-facto standard for coordinate representation are that:

- They have a better hardware support for higher resolutions. Given that double-precision IEEE754 floating-point numbers have a $52 + 1 + 1$ bits mantissa (implicit and sign bit included), they have resolution enough to represent any point in the Earth surface (which has a 40.000km equatorial perimeter) with a precision better than 3nm, a precision that should suffice for any GIS application[1]. The other data type well-supported by hardware (32 bits integers) would only allow a precision of 1cm.

- They are much better suited for automatically adapting the representation scale, thanks to the mantissa component.

The advantages of floating-point-based vectorial models are relevant enough for current general-purpose GIS applications, and the current state of the art on GIS relies in the strong interoperability between spatial technologies and data sources. Therefore, we should expect that spatial data providers will always use floating-point-based representations, and that all physical spatial models must be suited for accepting floating-point-based input data. This means that:

- A physical spatial vectorial model must be able to represent any point having IEEE754 floating-point coordinates.

- A physical spatial vectorial model must be able to represent any segment having end points with floating-point coordinates.

Even though one could select an appropriate Dualgrid to support the input of non-dualgrid fixed-size integer-based datasets, there is not such possibility for non-dualgrid datasets based on floating-point coordinates. To give support for importing datasets based on floating-point coordinates, we would have to choose a Dualgrid capable of representing any segment having end points with floating-point coordinates. The

[1] As far as we do not intend to map atoms.

simplest way to analyze it is to see floating-point numbers as a compact and lossy representation for large integers (counting very small units). According to the IEE754-2008 standard [IEEE08], 64 bits floating-point numbers have a $52+1+1$ bits mantissa (implicit and sign bits included) and a 10 bits exponent (ranging from -1022 to $+1023$). From a practical (and rough) point of view we can see it as a $54+1.023+1.022=2.099$ bits integer[2]. Applying Theorem 4.2.1 introduced in Chapter 4, it turn outs that the Dualgrid required would need rational coordinates with a $3*2.099+3=6.300$ bits numerator and a $2*2.099+3=4.201$ bits denominator. That is, each single point coordinate would require 10.501 bits. The size of rational numbers required for the coordinates in such a Dualgrid would be unacceptable.

5.1.2 Performance

The original Dualgrid proposal required using fixed precision rational coordinates which required approx. 5 times the space required by the original data based in integer coordinates. This imposed a sensible performance and storage toll.

In Section 4.4, we presented a partial adaptation of PostGIS v1.0.2 to Dualgrid representation that we implemented [Cre06]. Although the Dualgrid-PostGIS implementation showed that the adoption of Dualgrid allowed PostGIS and GEOS to get rid of their problems with robustness and consistency among operations, it also showed that the new implementation was in average 50 times slower than the original one (see Table 5.1[3]). Even if we consider further optimizations, one should expect at least a 20 times slowdown. The main reason is that the original Dualgrid proposal requires for all computations the use of finite precision numbers without hardware support. Even though the storage space is limited in size, the implementation cannot take advantage of the processor's hardware support for integer or IEEE754 floating-point computation. The Regular Polytope representation was proposed in [Tho07] using a discrete representation space with the same key logical properties as Dualgrid. The authors also make a proof of concept implementation where the computational cost of using fixed-size rational coordinates becomes evident.

[2]It would not be exactly that, as such an integer would be restricted to being allowed to have bits to 1 within a single consecutive block of 52+1 bits (the only ones stored in the mantissa). But it is a precise enough model to understand the space requirements for a Dualgrid supporting them as input coordinates.

[3]Tests with N/A values corresponds with tests for which PostGIS v1.0.2 failed with a *topology exception* error, so no execution times were recorded for them. In Table 4.1 (Section 4.4) we repeated the consistency tests for a newer version of PostGIS, v1.5.11. However, to do any performance test between Dualgrid-PostGIS (based in PostGIS v1.0.2) and PostGIS v1.5.11 would be useless and unfair, as the differences would not be only relative to using Dualgrid, but also to all the algorithm changes and optimizations between PostGIS v1.0.2 and PostGIS v1.5.11.Therefore, we keep here the original performance tests between PostGIS v1.0.2 and Dualgrid-PostGIS.

	Test	Times slower (Dualgrid-PostGIS vs PostGIS v1.0.2)
1	$(A \cap B) - A = \emptyset$	N/A
2	$(A \cap B) - B = \emptyset$	N/A
3	$(A \cap B) \subseteq A$	36,86
4	$(A \cap B) \subseteq B$	42,13
5	$A \cup (A \cap B) = A$	N/A
6	$B \cup (A \cap B) = B$	N/A
7	$(A \cap B) \subseteq (A \cup B)$	134,61
8	$A \subseteq (A \cup B)$	45,16
9	$B \subseteq (A \cup B)$	27,51
10	$disjoint((A-B),(A \cap B))$	47,29
11	$disjoint((B-A),(A \cap B))$	40,78
12	$disjoint((A-B),B)$	57,83
13	$disjoint((B-A),A)$	51,58
14	$(A \cup B) - B = (A-B)$	N/A
15	$(A \cup B) - A = (B-A)$	N/A
16	$(A-B) \subseteq A$	42,55
17	$B - A \subseteq B$	25,82
18	$(A \cup B) - (A \cap B) = symdiff(A,B)$	69,03
19	$(A-B) \cup (B-A) = symdiff(A,B)$	38

Table 5.1: Original PostGIS vs Dualgrid-PostGIS performance comparative.

5.2 Dualgrid For Floats

To adapt Dualgrid to the additional requirements imposed by the reality of current commercial spatial technologies, we will take into account the following premises:

1. Thanks to the incorporation of successful research in that area (e.g., [Mil89, DS90, ABD+97]), currently existing implementations of commercial spatial technologies already solve the problem of correctly testing (without error) the basic relations point-segment and segment-segment in IEEE754 floating-point-based representations, with acceptable performance. They are used to provide exact and consistent spatial predicates (e.g., *intersects*, *overlaps*, etc.), as well as to ensure that more complex algorithms take correct and consistent decisions on conditional tests.

2. Developers can accept the imposition of some additional restrictions to be fulfilled when constructing points and segments, as far as implementing and validating such restrictions become straightforward, intuitive and non-intrusive (that is, has no big impact on their algorithms).

3. The vast majority of the points that need to be explicitly represented in the result of a spatial operation are points that were already explicitly represented in the input arguments. Only a small percentage of them are new points resulting from boundary intersections between the input spatial objects. An evidence supporting this premise is provided in Section 5.3.

Taking all these premises into account, we introduce the Dualgrid For Floats (DualgridFF for short) physical representation model [CL10].

Definition 5.2.1. The *set of points* GP_F is the set of all possible points with floating-point coordinates.

Definition 5.2.2. The *set of segments* GS_F is the set of all the possible segments $S(P_{F1}, P_{F2})$ with end points in GP_F ($P_{F1} \in GP_F$, $P_{F2} \in GP_F$).

Definition 5.2.3. A *grid of points for floats* GP_{FF} is the set of points $GP_{FF} = GP_F \cup GP_I$, where GP_I is the set of all the points resulting of the intersection of segments GS_F.

Definition 5.2.4. A *grid of segments for floats* GS_{FF} is the set of segments $GS_{FF} = GS_F \cup GS_I$, where GS_I is the set of all possible segments defined over the line of any segment $S \in GS_F$ and resulting from shortening it by its intersection with other segments $S_i \in GS_F$.

Figure 5.1 shows some examples of elements belonging to the sets above:

- The grid represents the possible X and Y floating-point coordinates, and therefore all the points falling on the grid are GP_F points. The points P_{11}, P_{12}, P_{21}, P_{22}, P_{31}, P_{32}, P_{41} and P_{42}, drawn in the figure as black circles, are all GP_F points.

- All possible segments having end points in the grid are GS_F segments. The figure shows some of these possible GS_F segments drawn in dotted lines, namely segments (P_{11}, P_{12}), (P_{21}, P_{22}), (P_{31}, P_{32}) and (P_{41}, P_{42}).

- All possible intersections of GS_F segments are GP_I points. In the figure, GP_I points P_A, P_B and P_C are drawn as black squares.

- Finally, GS_I is the set of all possible segments resulting from shortening a GS_F segment by its intersections with other GS_F segments. For example, in Figure 5.1 we can identify the following GS_I segments:

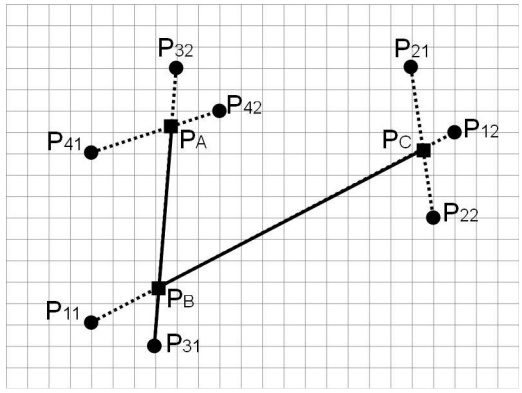

Figure 5.1: Example of points belonging to GP_F and GP_I and segments belonging to GS_F and GS_I.

- The GS_I segments (P_{11},P_B), (P_{11},P_C), (P_B,P_{12}) and (P_C,P_{12}), resulting from shortening GS_F segment (P_{11},P_{12}) with its intersections with the other drawn segments.
- The GS_I segments (P_{21},P_C) and (P_C,P_{22}), obtained from GS_F segment (P_{21},P_{22}).
- Tthe GS_I segments (P_{31},P_B), (P_{31},P_A), (P_B,P_{32}) and (P_A,P_{32}), obtained from GS_F segment (P_{31},P_{32}).
- The GS_I segments (P_{41},P_A) and (P_A,P_{42}), obtained from GS_F segment (P_{41},P_{42}).

From now on, and when speaking about GS_I segments, we will often use the notation $S = (S_{F1},S_{FS},S_{F2})$ to represent that S is the result of shortening the supporting GS_F segment S_{FS} by its intersections with GS_F segments S_{F1} (their intersection defines the S start point) and S_{F2} (their intersection defines the S end point).

Definition 5.2.5. A *Dualgrid for floats* (DualgridFF) DG_{FF} is defined as a pair of grids GP_{FF} and GS_{FF}, where GP_{FF} is a grid of points for floats and GS_{FF} is a grid of segments for floats. Formally speaking, $DG_{FF} = GP_{FF} \cup GS_{FF}$.

If the input spatial data are represented through vertexes with floating-point coordinates (points belonging to GP_F), their boundary segments will be segments belonging to GS_F. Therefore, the results of typical spatial operations on vectorial

models will be representable through those points and segments and points and segments resulting from the intersection of segments in GS_F (that is, points from GP_I and segments from GS_I). As a result, the DualgridFF defined above fulfills all the requirements for a *dualgrid* to accept input data with floating-point coordinates, and hence its use as the base for the representation of spatial objects ensures that all set-theory spatial operations remain closed.

For an efficient implementation of DualgridFF, and taking into account the premises previously described, we propose the following representation for the DualgridFF elements:

- Any point $P \in GP_F$ and any segment $S \in GS_F$ is represented as usual. For GP_F points that means that they are represented as a pair of floating-point coordinates. For GS_F segments that means representing them as the pair (P_{F1}, P_{F2}) defined by the segment end points.

- A point $P \in GP_I$ is represented as the pair of GS_F segments that define it, that is, as (S_{F1}, S_{F2}).

- In the representation of polylines and polygons (usually represented as sequences of points), each vertex in the sequence shares a common segment with the previous vertex and another common segment with the next vertex. This implies that for each vertex $P_I \in GP_I$ in the sequence of points (where $P_I = (S_{F1}, S_{F2}) = ((P_{11}, P_{12}), (P_{21}, P_{22}))$, P_{11} will correspond to the previous vertex in the sequence of points and P_{22} will correspond to the next vertex in the sequence of points. Therefore, to represent P_I in the sequence of points we only need to store a pair with the remaining two points, that is, the pair (P_{12}, P_{21})[4]. As a special case of polylines, GS_I segments are represented as it would correspond to a polyline with only two vertexes.

The proposed representation can be better understood with the help of Figure 5.2. On it, we can identify the following examples:

- Points drawn as black circles are GP_F points. They would be represented as usual, as a pair of floating-point coordinates.

- The simple polyline C is just one GP_F segment, so it would be represented as (P_{C1}, P_{C2}).

[4] Exceptions are first (respectively last) points of the sequence of points. As they do not have a previous (respectively next) point, an additional point needs to be represented for them (P_{11} if it is the first point in the sequence, P_{22} if it is the last one).

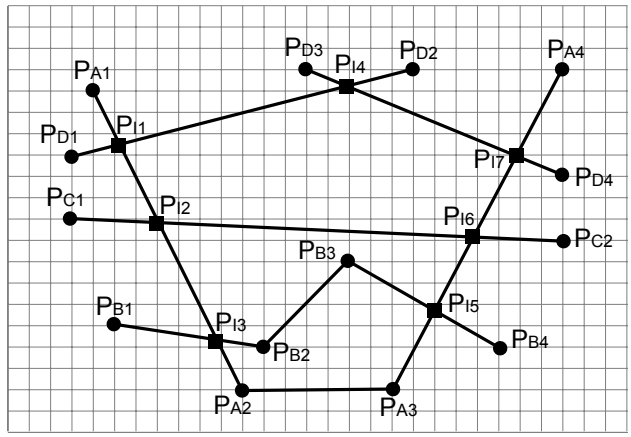

Figure 5.2: Examples of DualgridFF points and polylines.

- Points drawn as black squares are GP_I points (generated by the intersection of two segments). They would be represented as the pair of segments that define it. For example, P_{I2} would be represented as $((P_{A1}, P_{A2}), (P_{C1}, P_{C2}))$ and P_{I6} would be represented as $((P_{C1}, P_{C2}), (P_{A3}, P_{A4}))$.

- Polylines A and B have only GP_F vertexes. Therefore polyline A would be represented as $(P_{A1}, P_{A2}, P_{A3}, P_{A4})$ and polyline B as $(P_{B1}, P_{B2}, P_{B3}, P_{B4})$.

- Polyline D (the sequence of vertexes $(P_{D1}, ((P_{D1}, P_{D2}), (P_{D3}, P_{D4})), P_{D4})$) is a little bit more complex, as it contains the GP_I vertex P_{I4}. It would be represented as $(P_{D1}, (P_{D2}, P_{D3}), P_{D4})$.

- The following are a few more complex examples of polylines:

  - The polyline *A-B-A* starts in P_{A1}, continues along A until P_{I3}, then follows B until P_{I5} and then continues along A until point P_{A4}. It corresponds to the sequence of vertexes $(P_{A1}, ((P_{A1}, P_{A2}), (P_{B1}, P_{B2})), P_{B2}, P_{B3}, ((P_{B3}, P_{B4}), (P_{A3}, P_{A4})), P_{A4})$ and it would be represented as $(P_{A1}, (P_{A2}, P_{B1}), P_{B2}, P_{B3}, (P_{B4}, P_{A3}), P_{A4})$.
  - The polyline *A-C-A* starts in P_{A1}, continues along A until P_{I2}, then follows C until P_{I6} and then continues along A until point P_{A4}. It corresponds to the sequence of vertexes $(P_{A1}, ((P_{A1}, P_{A2}), (P_{C1}, P_{C2})),$

$((P_{C1}, P_{C2}), (P_{A3}, P_{A4})), P_{A4})$ and it would be represented as $(P_{A1}, (P_{A2}, P_{C1}), (P_{C2}, P_{A3}), P_{A4})$.

- The polyline *A-D-A* starts in P_{A1}, continues along A until P_{I1}, then follows D until P_{I7} and then continues along A until point P_{A4}. It corresponds to the sequence of vertexes $(P_{A1}, ((P_{A1}, P_{A2}), (P_{D1}, P_{D2})), ((P_{D1}, P_{D2}), (P_{D3}, P_{D4})), ((P_{D3}, P_{D4}), (P_{A3}, P_{A4})), P_{A4})$. It would be represented as $(P_{A1}, (P_{A2}, P_{D1}), (P_{D2}, P_{D3}), (P_{D4}, P_{A3}), P_{A4})$.
- The subset of polyline A that follows A from point P_{I1} to point P_{I6} has the peculiarity that its first and last vertexes are GP_I points. It corresponds to the sequence of vertexes $(((P_{D1}, P_{D2}), (P_{A1}, P_{A2})), P_{A2}, P_{A3}, ((P_{A3}, P_{A4}), (P_{C1}, P_{C2})))$ and it would be represented as $(((P_{D1}, P_{D2}), P_{A1}), P_{A2}, P_{A3}, (P_{A4}, (P_{C1}, P_{C2})))$.
- A particular case of a polyline with first and last vertexes being GP_I points is the GS_I segment (P_{I2}, P_{I6}). It corresponds to the sequence of vertexes $(((P_{A1}, P_{A2}), (P_{C1}, P_{C2})), ((P_{C1}, P_{C2}), (P_{A3}, P_{A4})))$ and it would be represented as $(((P_{A1}, P_{A2}), P_{C1}), (P_{C2}, (P_{A3}, P_{A4}))$. Another particular case is the GS_I segment (P_{I2}, P_{C1}). It results from shortening segment (P_{C1}, P_{C2}) only on one of its end points (on its intersection with segment (P_{A1}, P_{A2})). It corresponds to the sequence of vertexes $(((P_{A1}, P_{A2}), (P_{C1}, P_{C2})), P_{C2})$ and it would be represented as $(((P_{A1}, P_{A2}), P_{C1}), P_{C2})$.

With the proposed representation, and with the assumptions stated in Section 5.2, almost all represented points will belong to GP_F and computations involving them will have no significant storage or performance penalties (evidence of this is provided later at Section 5.3.1). This will make it worth to have different representations for GP_F and GP_I points, as the additional complexity introduced will be compensated by being able to compute using hardware floating-point arithmetic in almost all cases. As a result, the average storage requirements and performance will be close to the ones provided by currently existing implementations.

With regard to the implementation of algorithms and the additional resolution required for intermediate results, we want to point out that in the end GP_Ivalues and GS_Ivalues are represented by their originating GP_F and/or GS_F parts, and hence predicates analyzing their relationships can be computed using the existing algorithms (described in the first premise presented above) that check whether two segments intersect or whether a point is on a segment. This is explained in detail later in Section 5.3.2.

Finally, an additional advantage of the proposed representation is that it intuitively forces the user to fulfill all the additional restrictions imposed by Dualgrid. For

example, a GP_I point P_I should be represented by two intersecting GS_F segments. With other representations, the need to fulfill such requirement could seem arbitrary and problematic because it could happen that such a point is in fact resulting from the intersection of two GS_I segments GS_{I1} and GS_{I2}. However, as the representation of a GS_I segment contains its supporting GS_F segment, it is straightforward to represent P_I by the supporting segments of GS_{I1} and GS_{I2}. The same happens when trying to represent a segment GS_I resulting of cutting it at its intersection with another GS_I segment. Hence, the restrictions on the objects that can be represented and the ways of addressing and fulfilling them become intuitive when using the proposed representation.

5.3 Implementation issues

In the previous section, we defined DualgridFF and we proposed a representation that reduces its storage and performance impact. In this section we will explain the factors that, combined with such a representation, will allow us to keep such a low impact. Moreover, we will suggest some tips that will help developers to reduce even more the penalty of using a DualgridFF representation.

5.3.1 Storage and performance cost of DualgridFF

First of all, it is necessary to prove the assumption that, from the collection of points that need to be explicitly represented when representing spatial objects, only a small percentage will belong to GP_I and therefore will need high resolution non-floating-point coordinates. For that purpose, we performed the following test. The data collection used for the tests was the same collection used (and described) in the tests of Section 4.4.

In this test we measured how many points are used to represent the geometries resulting of spatial operations and which percentage of them correspond with points not present in the input geometries (and hence, they are generated by the spatial operations). With that purpose, we used PostGIS to compute the intersection, union, difference and symdifference of all overlapping geometries in both tables (`comarcas` and `municipios`).

The results of the test can be seen on Table 5.2. The first column with the number of points shows the number of that type of points in the original datasets. The second column shows the number of that type of points in a collection with the original datasets plus all the possible intersections among objects on those datasets. And finally the third column shows the number of that type of points in a collection with the

Collection	original datasets	original datasets + ∩	original datasets + ∩ + ∪ + difference + symdifference
Unique points	111.444	*113.794*	113.794
Unique points not present in original collection	–	*2.350*	2.350
% different new points	–	2,066%	2,066%
Points stored (duplicates included)	241.418	*1.020.654*	14.386.484
Stored points not present in original collection (duplicates included)	–	*3.379*	22.492
% new points stored	–	0,331%	0,156%

Table 5.2: Percentage of new points generated by spatial operations.

original datasets plus all the possible intersections, unions, differences and symmetric-differences among objects on those datasets. The first row shows the number of unique points (ignoring duplicates). The second row shows how many of those unique points did not existed in the original datasets (that is, are new points computed during the spatial operations). The third row shows the percentage of the total points that are new computed points. Rows fourth to sixth show the same information, but this time counting the number of points effectively stored, (that is, without removing duplicated points).

The last row of Table 5.2 evidences the impact of the proposed GP_I points (new points) representation in performance and storage requirements. The test shows that after inserting the intersection results only 0,331% of the points represented in the geometries stored are GP_I points.

If we extrapolate the numbers to estimate the storage penalty of DualgridFF, the storage requirements would be $4 \times 0,00331 + (1 - 0,00331) = 100,993\%$ of non-DualgridFF models. That is, the storage impact would be less than 1%. With regard to performance penalty, and supposing that computations involving GP_I points are 50 times slower than computing with GP_F points (a typical penalty when using arbitrary precision computation), that would mean a computational cost of $50 \times 0,00331 + (1 - 0,00331) = 116,219\%$ over non-DualgridFF models, that is, a 16% performance penalty. We believe that these storage and performance cost are acceptable in exchange of consistency in all operations.

5.3.2 Performance improving tips

With regard to computations related to GP_I points and GSP_I segments, there are several tips that would probably further reduce the performance penalty. Most of them are directly linked to the advantages of adopting the element representation proposed in Section 5.2.

Point-segment orientation test: One of the more typical operations in geometrical computation algorithms is the *point-segment orientation test*, that tests whether a point P is to the left, to the right or in a segment S. It is used for three main purposes: 1) Testing if $P \in S$, 2) testing if two segments S_1 and S_2 intersect (which is implemented as a set of tests checking the orientation of the end points of one segment with respect to the other segment) and 3) testing whether a point is inside a polygon. The possible cases in DualgridFF are:

- $P \in GP_F$ and $S \in GS_F$: This is the traditional floating-point-based point-segment orientation test, usually implemented as set of exact 2x2 determinant sign tests [ABD+97].
- $P \in GP_F$ and $S \in GS_I$: In this case, $S = (S_{F11}, S_{FS}, S_{F22})$, where S_{FS} is the supporting segment. For the point-segment orientation test, we should perform the orientation test between P and S_{FS} using the traditional tests. If the orientation test returns that P is in the line of the supporting segment S_{FS}, we need to test P with $S_{F11}$and S_{F22} to answer the original test. Therefore, the cost of this test would be approx. 3 times the cost for GS_F segments.
- $P \in GP_I$: Given the cases in which a GP_I point $P = (S_{F1}, S_{F2})$ is generated, almost in all cases when a point $P \in S$ it is because it was generated by an intersection between S and another segment. Therefore, just testing whether S is one of the originating segments of P will detect (almost) all positive answers. If $S \in GS_I$ ($S = (S_{F11}, S_{FS}, S_{F22})$) the test should be performed against its supporting segment S_{FS}. This comparison should be even faster than the traditional *point_in_segment* test. If the test fails, most answers can be computed using the approximated floating-point value of P and performing the point-segment orientation test if its distance to S is bigger than a *error threshold* (which ensures that computing rounding errors cannot lead to a wrong answer). Only cases not decided by these tests should be computed using either arbitrary precision computation or a *three segments intersection* test. A pseudocode is shown in Algorithm 5.1.

Point comparison: Another of the fundamental tests in geometrical algorithms is whether two points are equal ($P_1 = P_2$). The possible cases are:

Algorithm 5.1 Point-segment orientation test when $P \in GP_I$.

```
// Notations: Semantics of the results of point_segment_orientation_test:
//          0 -> P in S
//          positive number ->  P to the left of S
//          negative number ->  P to the right of S
//
// Remember, P in GP_I implies that P=(S_F1, S_F2)

if (S in GS_F) then
    if (S=P.S_F1 or S=P.S_F2) then
        // S is an originating segment of P
        return 0 // P in S
    end
else // S in GS_I, thus S=(S_F11, S_FS, S_F22)
    if (S.S_FS=P.S_F1 or S.S_FS=P.S_F2) then
        // S.S_FS is an originating segment of P
        return 0 // P in S
    end
end
// If none of the previous tests resolved
if (distance(approx(P),S) > THRESHOLD) then
    // THRESHOLD ensures that approximated computation is correct.
    return floatpoint_orientation_test(approx(P),S) // Cases 1 and 2
else
    // Test needs to be computed with exact computation
    return EXACT_orientation_test(P,S)
end
```

- P_1 and P_2 are GP_F points. This test is a simple comparison of floating-point coordinates.

- $P_1 \in GP_F$ and $P_2 \in GP_I$. As $P_2 = (S_{F1}, S_{F2})$, this test can be performed with two *point in segment* tests, checking whether $P_1 \in S_{F1}$ and $P_1 \in S_{F2}$.

- P_1 and P_2 are GP_I points. Almost all positive cases ($P_1 = P_2$) will be caused by P_1 and P_2 being originated from the intersection of the same pair of segments. Therefore, comparing their representation segments will answer the test. Most of negative cases ($P_1 \neq P_2$) can be discarded by computing the approximated floating-point values of $P_1$and P_2 and testing whether their distance to S is bigger than a *error threshold*. Only cases not decided by these tests should be computed using either arbitrary precision computation or two *three segments intersection* tests. The pseudocode for this part is shown in Algorithm 5.2.

Algorithm 5.2 Point_comparison algorithm when P_1 and P_2 are GP_I points.

```
// Results:
//    TRUE  -> P1 equalTo P2
//    FALSE -> P1 NotEqualTo P2
// Remember, P_1 in GP_I, P_2 in GP_I
// NOTE: Assuming P=(S_F1, S_F2) sets an order between S_F1 and S_F2

if ((P1.S_F1 == P2.S_F1) and (P1.S_F2 == P2.S_F2)) then
    return TRUE
else
    if distance(approx(P1), approx(P2)) > THRESHOLD then
        // THRESHOLD is the biggest possible error due to approximation
        return FALSE
    else
        EXACT_PointComparison(P1, P2)
    end
end
```

As we can see, the proposed representation of the DualgridFF elements has the additional advantage of allowing us to avoid the use of arbitrary precision computation in the vast majority of cases. For points not originated from the involved segments, the *distance to segment* filter should decide almost all negative answers (and, in fact, in this case the vast majority of tests will have a negative answer). For points originated from the involved segments (which should account for the vast majority of positive answers), the comparison with the originating segments should decide the positive answer. The use of *three segments intersection* tests or performing arbitrary precision computations will only be needed in the remaining cases.

5.3.3 Interoperability

DualgridFF has already been defined in such a way that input interoperability (that is, the capability to represent data generated by currently existing applications) is guaranteed (it was in fact one of the driving reasons for its definition). Output interoperability with external applications (feeding them with data generated from a DualgridFF-based spatial database) can be achieved in two ways:

- For non-DualgridFF-aware applications or that do not require consistency among operations (e.g., visualization applications) it will suffice to return approximate values for the GP_I points. Therefore, exchange formats as GML, or WKB and WKT defined by SFS [OGC06] will be suitable.

- For DualgridFF-aware applications that require to consistently process its data (and hence need to input or output DualgridFF-based data), the main problem is that current exchange formats are (obviously) not designed for the DualgridFF representation. However, the nature of the GML format (XML-based), and the fact that GML specifically allows its extension with personalized tags could allow to generate (and accept) GML-compatible files extended with DualgridFF information, so that non-DualgridFF-aware applications could process them as traditional GML files, and DualgridFF-aware applications could make use of that additional DualgridFF information. For example, when a DualgridFF spatial object (which can have GP_I points in its boundary representation) is exported to a GML-compatible file, it can be represented as a typical GML spatial object (using the floating-point coordinate approximation for the points that are GP_I points), and extended with an additional attribute that, for each point $P \in GP_I$ (with $P = (S_{F1}, S_{F2})$) in the vertexes list, indicates the point index in the vertexes list and its two originating segments S_{F1} and S_{F2}.

5.4 Conclusions

In this chapter we presented *Dualgrid For Floats*, a physical spatial data representation model that is designed to be used with floating-point coordinates, thus satisfying the requirement of interoperability with existing technology and datasets. Furthermore, DualgridFF succeeds in keeping the theoretical properties of the abstract spatial data models (i.e., closure under set-theory operations) while not degrading the performance and the storage requirements of the implementation. It also keeps all the properties exhibited by Dualgrid. Hence, any reference to the consistency among operations of Dualgrid in Table 4.2 can be read as a reference to DualgridFF as well.

Chapter 6

Conclusions and future research lines

6.1 Summary of contributions

The problem of defining and implementing a physical spatial data model that maintains the properties defined by an abstract spatial data model is still open. Current implementations of spatial database technology fail to maintain such properties, and this generates problems of inconsistencies among operations (Appendix A shows a very simple example that produces inconsistent answers in three main commercial spatial DBMS, namely Oracle Spatial, PostgreSQL/PostGIS and Microsoft SQL Server). In some cases, this problem even causes sporadic unexpected crashes and exceptions in their geometrical algorithms (as Section 4.4 evidences with tests performed to PostgreSQL/PostGIS, and as can be read in Section 1.5.5 of *Oracle Spatial Developer's Guide* [Ora10] with respect to the tolerances machinery of Oracle Spatial). As a result, GIS software developers and GIS users are imposed the burden of dealing with the consequences of a broken algebra[1].

There have been research proposals that solve this problem at the level of the discrete spatial data model [GS93], but they do not completely solve the problem, and their drawbacks (as explained in Section 2.4.2) prevent them from being used as a solution for commercial spatial DBMS. This research work has focused on covering

[1] We use the term *broken algebra* to remark that using an implementation that no longer fulfills the properties of the abstract model undermines all the efforts devoted to its definition because the users can no longer rely on the mathematical and logical grounds of the abstract model.

the existing gap between the research work on discrete spatial data models and the implementations in commercial spatial database management systems and GIS tools. Therefore, this thesis makes three major contributions:

- A detailed study of the state of the art and an analysis of the requirements that a representation space for vectorial models should fulfill to guarantee that it remains closed under the main spatial operations, specially set-theory operations.
- The proposal of a new representation space (Dualgrid) that fulfills those requirements, and therefore ensures that the properties of the corresponding abstract model are kept, allowing the exact implementation of the operations (instead of approximated implementations of them) whereas using fixed-size coordinates.
- The proposal of DualgridFF, an evolution of Dualgrid towards a physical spatial model designed to fulfill the additional requirements in terms of performance and interoperability of current professional users[2].

Regarding the analysis of the problem, Chapter 3 presented a comparison of the consistency properties of the spatial operations of the more common spatial data models (with emphasis on vectorial data models and the approaches used on them to address the discretization problems).

Chapter 4 identifies the properties that a physical spatial data model needs to fulfill to be able to implement an abstract/discrete spatial data model without breaking its closure properties. Then, it defines Dualgrid, a new fixed-size discrete representation space that succeeds in defining a physical spatial data model suitable to be used on computers and keeping all the key properties of the discrete spatial models, allowing the exact implementation of their operations, instead of approximations of them. Also, Section 4.2 analyzes Dualgrid in terms of interoperability, showing what considerations should be taken into account to guarantee that existing external datasets can be imported in the system exactly.

We have also shown its capability to be adopted by already implemented spatial algorithms because it requires no changes to their logic. Sections 4.3 and 4.4 show its effectiveness and versatility with the adaptation of two different existing spatial implementations to the use of Dualgrid as its representation space. Sections 4.3 shows the results of adapting an existing implementation of the ROSE algebra [GS95], using Dualgrid as the representation space for the underlying Realm [GS93, MPF+96] and without requiring the modification of any of the ROSE algebra algorithms [GdRS95]. Section 4.4 shows the results of an experimental adaptation of PostGIS-GEOS (the

[2]We use here the term *professional users* in contraposition to *academic researchers*.

widely used spatial extension of PostgreSQL), proving that the use of Dualgrid as the underlying representation space not only solves the problems of inconsistencies among operations that PostGIS-GEOS exhibits, but also all its usual *topology exception* problems when combining spatial operations.

Finally, Chapter 5 identifies additional needs of the GIS community that must be successfully covered in order to provide spatial databases technologies suitable for commercial use (e.g., accept input data using floating-point coordinates and low computational and storage overhead). These new requirements are taken into account in the definition of DualgridFF (*Dualgrid For Floats*), an evolution of the Dualgrid discrete space proposal towards a physical spatial data model. DualgridFF represents a trade-off between the needs of researchers, spatial databases/technologies developers, application developers and final users. It provides a solid physical representation model that succeeds in keeping the main properties of the original abstract and discrete vectorial models, so that spatial databases and technologies implementors can easily translate them to their implementations, whereas keeping their behaviors intuitive and well-defined.

Dualgrid and DualgridFF provide a solid basis to solve the robustness and consistency problems of current spatial vectorial technologies by attacking the roots of the problems. They provide the tools to keep the closure properties of abstract data models at the physical data model level. Moreover, DualgridFF goes a step further by taking into account the needs of all spatial technology stakeholders, so that spatial databases technology implementations based on it can successfully conciliate their needs, helping to join efforts in the evolution of geographic information systems.

6.2 Future work

This research work has made a big step forward in providing spatial databases technologies with the grounds for implementing robust and consistent spatial data models. As a result of it, the following future research lines can be identified:

- *Improvement of spatial data management libraries.* The fastest way to translate the advantages of the proposals of this research work to real world GIS applications is to improve existing and widely used spatial data management libraries by making them DualgridFF capable. A clear candidate is the JTS Topology Suite (`http://sourceforge.net/projects/jts-topo-suite/`), a library widely used by GIS frameworks, tools and applications to provide some spatial analysis capabilities. Making JTS DualgridFF capable (adding a DualgridFF precision layer to the ones already existing) would allow GIS

application developers to simplify their spatial analysis implementations and to evolve them to more advanced analysis levels, as the DualgridFF physical data model will guarantee them that their spatial analysis algorithms can rely on consistent and robust spatial operations. Moreover, such an adaptation of JTS would allow to do an experimental validation of the estimated performance and storage costs of using DualgridFF and the optimization techniques proposed in [CL10]. It would also let us verify its effectiveness and ease of use for adapting existing implementations.

- *Improvement of spatial databases extensions*. A second (but not less relevant) path would be to evolve existing spatial database extensions to adapt them to the DualgridFF physical data model. It would allow them to provide consistent operations, as well as (not less important) to free them from all the algorithmic complexities they currently implement to try to (just) minimize the robustness problems that their current non-closed discrete space generate. As a result, robustness should improve and internal implementations should simplify, making the future implementation of more powerful spatial operations easier. Additionally, it would allow us to further analyze implementation strategies in order to optimize performance and storage requirements, and reduce the impact on incorporating DualgridFF to existing implementations. A candidate for such an evolution is PostGIS/GEOS [Ref10], the open source spatial database extension used by PostgreSQL.

- *Optimization of arbitrary precision libraries to DualgridFF specific requirements*. Another future research line would be to analyze algorithms for improving arbitrary precision computations in the specific cases that they are required in DualgridFF. Given that such arbitrary precision computations would be performed in very specific scenarios, it is very likely that an specifically designed library would exhibit a much better performance for a DualgridFF implementation.

- *Development of more advanced spatial analysis support in existing GIS frameworks*. Our research group has a large experience on the development of advanced GIS tools. Development of spatial analysis support on them has been always difficult, due to the limitations imposed by the inconsistency among operations of current spatial technologies. The availability of DualgridFF based libraries would solve those limitations, opening the opportunity of an evolution on the quality and power of the spatial analysis support provided within GIS applications.

- *Extension to spatiotemporal models.* This thesis has been focused in non-temporal spatial models. A lot of research has been devoted in the last decade to the development of spatiotemporal data models and algorithms [CFG+03], where the temporal domain of spatial information is also taken into account. Depending on the model, the temporal evolution of spatial data can be either discrete (the spatial value changes at specific time instants) or continuous (where the spatial values evolve continuously over the time, usually called in the literature *moving objects*). The use of Dualgrid and DualgridFF for the implementation of discrete spatiotemporal models should be relatively straightforward. The partial implementation in [CFG+03] was already prepared to be used with spatial data following the Dualgrid restrictions, and when static slices were used (when the spatial value represented by the slice was static in all the time interval) and Dualgrid restrictions were fulfilled, the implementation was already able to provide consistency among spatial and spatiotemporal operations.

  The application of Dualgrid and DualgridFF to fully moving objects, however, needs a deeper study. It should be possible to extend the Dualgrid approach to 3D geometries (where the third dimension would be used for the time axis), and to represent the spatial evolution of an object for each time slice as one such a 3D geometry. In fact, in [Tho07] they already define a 3D representation (being the height the third dimension) based on the lessons from Dualgrid. This would help to keep consistency among spatiotemporal operations on continuously moving objects. However, the difficult part would be to provide consistency among spatial and spatiotemporal operations. The reason is that spatiotemporal models need to provide support to projections on the time dimension (e.g., to retrieve the time when a spatiotemporal object occupied a given position, or computing the time span covered by the spatiotemporal object) and on the space dimensions (e.g., to retrieve the spatial value of a spatiotemporal object at a given time instant, or the projection of the trajectory of the entire spatiotemporal object onto the space). Hence, an evolution of Dualgrid to the spatiotemporal domain (let us call it here DualgridST) would at least have to define a representation space such that projections onto the space of spatiotemporal values would still be closed under the representation space. Additionally, it should ideally be able to handle the combination of projections onto the space with projections onto the time. If these combinations were not handled appropriately, sequences of these operations would very likely generate spatial values that would not conform to the DualgridST. For example, suppose that we have the fully moving objects ST_1, ST_2 and ST_3 and the spatial value S_1. If we query the time instant T_1 at which $S_1 = ST_1$, and then we ask the spatial value that ST_2 takes at time T_1 we get S_2.

S_2 will very likely require for its representation higher precision than S_1. If we now ask for the time instant T_2 at which $S_2 = ST_2$, it will very likely require more precision than the one required by T_1. If we now ask the spatial value S_3 that ST_3 takes at time T_2, it will very likely require even higher precision than S_2. This is, the successive combination of time and spatial projections can escalate the precision required for representing temporal and spatial values. This is a problem that should be carefully analyzed and addressed when extending Dualgrid to spatiotemporal models.

Appendix A

Spatial inconsistencies example

This appendix shows a simple example that highlights how easy is to get inconsistent spatial answers from current commercial spatial DBMS (Oracle Spatial, PostgreSQL/PostGIS and Microsoft SQL Server).

The example to test the behavior of those DBMS is shown in Figure A.1. On it, two simple overlapping geometries (a rectangle *region*1 and a triangle *region*2) are shown. Both of them are represented through their vertex, all of them having small integer coordinates. However, their boundaries intersect at some points, $(2, 7/3)$ and $(6, 11/3)$, having coordinates not exactly representable with finite-size floating-point numbers. The test is simple: to compute the intersection between both geometries, and query the DBMS whether their intersection (named $1intersection2$) is contained in each of the originating geometries. According to the set-theory properties of intersection, the resulting geometry $1intersection2$ should be contained in both *region*1 and *region*2.

The following sections show that all of them (Oracle Spatial in Section A.1, PostgreSQL/PostGIS in Section A.2 and SQL Server in Section A.3) fail to pass such a simple test.

A.1 Intersection test in Oracle Spatial

Section A.1.1 shows the SQL sentences used to test the example under Oracle Spatial 11g. Once both geometries and their intersection are inserted in the table, the table content is the one shown in Table A.1. Table A.2 shows the result of the test for each combination of the geometries. Note that in this case the test has been implemented returning TRUE if either *SDO_EQUAL()*, *SDO_COVERS()* or *SDO_CONTAINS()*

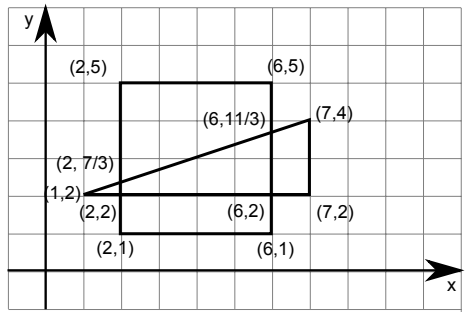

Figure A.1: Simple set-theory test.

returns TRUE. The reason for it is that Oracle implements such operations in a strictly disjoint semantic, so that only the one of them that better fits to the relationship returns true. For example, if two geometries are equal, *SDO_EQUAL()* will return true, and hence *SDO_COVERS()* and *SDO_CONTAINS()* will return false.

MY_ID	MY_NAME	SDO_UTIL.TO_WKTGEOMETRY(MY_SHAPE)
1	region1	POLYGON ((2.0 1.0, 6.0 1.0, 6.0 5.0, 2.0 5.0, 2.0 1.0))
2	region2	POLYGON ((1.0 2.0, 7.0 2.0, 7.0 4.0, 1.0 2.0))
12	1intersection2	POLYGON ((2.0 2.33333333333333, 2.0 2.0, 6.0 2.0, 6.0 3.66666666666667, 2.0 2.33333333))

Table A.1: Table test_regions in Oracle after inserting both geometries and their intersection.

A.1.1 Oracle commands

```
-- 1 - Table creation:
-- ------------------
CREATE TABLE test_regions (
   my_id integer PRIMARY KEY,
   my_name VARCHAR(64),
   my_shape SDO_GEOMETRY);

-- 2 - Geometries insertion:
-- ------------------------
-- 2.a) A rectangle:
INSERT INTO test_regions VALUES(
   1,
```

A_NAME	B_NAME	EQUALS or COVERS or CONTAINS
region1	region1	TRUE
region1	region2	FALSE
region1	1intersection2	TRUE
region2	region1	FALSE
region2	region2	TRUE
region2	1intersection2	FALSE * (Should have returned TRUE)
1intersection2	region1	FALSE
1intersection2	region2	FALSE
1intersection2	1intersection2	TRUE

Table A.2: Oracle answers to the contains test.

```
  'region1',
  SDO_GEOMETRY(
    2003,  -- two-dimensional polygon
    NULL,
    NULL,
    SDO_ELEM_INFO_ARRAY(1,1003,3),  -- one rectangle (1003 = exterior)
    -- only 2 points needed to define a rectangle (lower left and
      -- upper right) with Cartesian-coordinate data
    SDO_ORDINATE_ARRAY(2,1, 6,5)
  )
);

-- 2.b) A triangle:
INSERT INTO test_regions VALUES(
  2,
  'region2',
  SDO_GEOMETRY(
    2003, -- 2D polygon
    NULL,
    NULL,
    SDO_ELEM_INFO_ARRAY(1,1003,1), -- one polygon (exterior ring)
    SDO_ORDINATE_ARRAY(1,2, 7,2, 7,4, 1,2)
  )
);

-- 3 - Adding metadata and index (required by Oracle Spatial):
-- ------------------------------------------------------------
-- 3.a) Metadata insertion
INSERT INTO user_sdo_geom_metadata (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
  VALUES (
    'TEST_REGIONS',
    'MY_SHAPE',
    SDO_DIM_ARRAY(  -- 20X20 grid
      SDO_DIM_ELEMENT('X', 0, 20, 0.00000000000000000001),
      SDO_DIM_ELEMENT('Y', 0, 20, 0.00000000000000000001)
    ),
  NULL   -- SRID
```

```
);

-- 3.b) Index creation (R-tree)
CREATE INDEX indice_spatial_idx ON test_regions(my_shape)
  INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- 4 - Intersection computation:
-- ----------------------------
-- NOTE: Boundaries intersect at points (2, 2), (6, 2), (6, 11/3) and (2, 7/3).
INSERT INTO test_regions
  SELECT
    '12',
    'lintersection2',
    SDO_GEOM.SDO_INTERSECTION(a.my_shape, b.my_shape, 0.0000000000000000001)
  FROM test_regions a, test_regions b
  WHERE a.my_name = 'region1' AND b.my_name = 'region2';

-- 5 RESULTS:
-- ----------
-- 5.a) Table content
SELECT my_id, my_name, SDO_UTIL.TO_WKTGEOMETRY(my_shape)
FROM test_regions;

-- MY_ID MY_NAME          SDO_UTIL.TO_WKTGEOMETRY(MY_SHAPE)
-- ----- ---------------- ---------------------------------------------------------------
-- 1     region1          POLYGON ((2.0 1.0, 6.0 1.0, 6.0 5.0, 2.0 5.0, 2.0 1.0))
-- 2     region2          POLYGON ((1.0 2.0, 7.0 2.0, 7.0 4.0, 1.0 2.0))
-- 12    lintersection2   POLYGON ((2.0 2.33333333333333, 2.0 2.0, 6.0 2.0,
--                                  6.0 3.66666666666667, 2.0 2.33333333))

-- 5.b - Containment tests:
-- NOTE: In Oracle Contains/Covers/Equals are strictly defined (only one is hold),
-- in contrast to PostGIS or SLQLServer.
SELECT a.my_name AS a_name, b.my_name AS b_name,
  CASE WHEN (
    SDO_COVERS(a.my_shape, b.my_shape)='TRUE' OR
    SDO_CONTAINS(a.my_shape, b.my_shape)='TRUE' OR
    SDO_EQUAL(a.my_shape, b.my_shape)='TRUE'
  ) THEN 'TRUE' ELSE 'FALSE' END AS Contains
FROM test_regions a, test_regions b ;

-- A_NAME           B_NAME           CONTAINS
-- ---------------  ---------------  --------
-- region1          region1          TRUE
-- region1          region2          FALSE
-- region1          lintersection2   TRUE
-- region2          region1          FALSE
-- region2          region2          TRUE
-- region2          lintersection2   FALSE * (* Should have returned TRUE)
-- lintersection2   region1          FALSE
-- lintersection2   region2          FALSE
-- lintersection2   lintersection2   TRUE
```

A.2 Intersection test in PostgreSQL/PostGIS

Section A.2.1 shows the SQL sentences used to test the example under PostgreSQL 9.1 with PostGIS 1.5. Table A.3 shows the content of the test table once both geometries and their intersection are inserted. Table A.4 shows the result of testing, for each combination of the geometries, whether the first contains the second. In this case, PostGIS spatial predicate *ST_COVERS()* is used, as its semantic is (slightly) more appropriate for our test than the one provided by *ST_CONTAINS()*.[1]

MY_ID	MY_NAME	ST_ASText(MY_SHAPE)
1	region1	POLYGON((2 1, 6 1, 6 5, 2 5, 2 1))
2	region2	POLYGON((1 2, 7 2, 7 4, 1 2))
12	1intersection2	POLYGON((6 3.66666666666667, 6 2, 2 2, 2 2.33333333333333, 6 3.66666666666667))

Table A.3: Table test_regions in PostgreSQL/PostGIS after inserting both geometries and their intersection.

A_NAME	B_NAME	A_COVERS_B
region1	region1	TRUE
region1	region2	FALSE
region1	1intersection2	TRUE
region2	region1	FALSE
region2	region2	TRUE
region2	1intersection2	FALSE * (Should return TRUE)
1intersection2	region1	FALSE
1intersection2	region2	FALSE
1intersection2	1intersection2	TRUE

Table A.4: PostgreSQL/PostGIS answers to the contains test.

A.2.1 PostgreSQL/PostGIS commands

```
0 - Database creation:
--------------------
-- Inherits from template "postgis" to be able to use PostGIS support
```

[1] Anyway, and for this specific example, both of them would return exactly the same answers.

```
CREATE DATABASE "test-postgis"
  WITH ENCODING='UTF8'
    TEMPLATE=template_postgis
    CONNECTION LIMIT=-1;

-- NOTE: Remember to switch to the newly created database from this point on

-- 1 - Table creation:
-- ---------------------
-- 1.a) Creating a normal (non spatial) table:
CREATE TABLE test_regions (
  my_id integer PRIMARY KEY,
  my_name VARCHAR(64)
  );

-- 1.b) Adding a geometry column:
SELECT AddGeometryColumn(
  'test_regions',
  'my_shape',
  -1,
  'GEOMETRY',
  2
  );

-- 2 - Geometries insertion:
-- ---------------------------
-- 2.a) A rectangle:
INSERT INTO test_regions (my_id, my_name, my_shape) VALUES (
  1,
 'region1',
 ST_GeomFromText('POLYGON((2 1, 6 1, 6 5, 2 5, 2 1))',-1));

-- 2.b) A triangle:
INSERT INTO test_regions (my_id, my_name, my_shape) VALUES (
  2,
  'region2',
  ST_GeomFromText('POLYGON((1 2, 7 2, 7 4, 1 2))',-1));

-- 3 - Intersection computation:
-- -------------------------------
-- NOTE: Boundaries intersect at points (2, 2), (6, 2), (6, 11/3) and (2, 7/3).
INSERT INTO test_regions
  SELECT '12', '1intersection2', ST_INTERSECTION(a.my_shape, b.my_shape)
  FROM test_regions a, test_regions b
  WHERE a.my_name = 'region1' AND b.my_name = 'region2';

-- 4 RESULTS:
-- ------------
-- 4.a) Table content
SELECT my_id, my_name, ST_ASText(my_shape) FROM test_regions;

-- MY_ID  MY_NAME         ST_ASText(MY_SHAPE)
-- -----  --------------  ---------------------------------------------------------------
-- 1      "region1"       "POLYGON((2 1, 6 1, 6 5, 2 5, 2 1))"
-- 2      "region2"       "POLYGON((1 2, 7 2, 7 4, 1 2))"
-- 12     "1intersection2" "POLYGON((6 3.66666666666667, 6 2, 2 2,
--                                 2 2.33333333333333, 6 3.66666666666667))"
```

```
-- 4.b - Containment tests:
SELECT a.my_name AS a_name, b.my_name AS b_name,
  ST_COVERS(a.my_shape, b.my_shape) AS a_covers_b
  FROM test_regions a, test_regions b ;

-- A_NAME          B_NAME          A_COVERS_B
-- --------------  --------------  ----------
-- "region1"       "region1"       t
-- "region1"       "region2"       f
-- "region1"       "1intersection2" t
-- "region2"       "region1"       f
-- "region2"       "region2"       t
-- "region2"       "1intersection2" f  * (* Should return TRUE)
-- "1intersection2" "region1"      f
-- "1intersection2" "region2"      f
-- "1intersection2" "1intersection2" t
```

A.3 Intersection test in SQL Server

Section A.3.1 shows the command used to test the example under Microsoft SQL Server 2012 RC0. Table A.5 shows the test table content after inserting both geometries and their intersection. Table A.6 shows the result of the test for each combination of the geometries, whether the first contains the second. In this case, SQL Server spatial predicate *STContains()* is used.

Note that in this case weird effects appear in the coordinates of the computed intersection. Some vertex whose X coordinates would have been expected to be directly copied from those in the input coordinates (as *region1* is a rectangle with boundaries perpendicular to the axis), have been somehow slightly moved. This could have increased the inconsistencies of the test results, if the X coordinate movement had resulted to move the affected points outside of *region1* instead of keeping them inside.

MY_ID	MY_NAME	ST_ASText(MY_SHAPE)
1	region1	POLYGON ((2 1, 6 1, 6 5, 2 5, 2 1))
2	region2	POLYGON ((1 2, 7 2, 7 4, 1 2))
12	1intersection2	POLYGON ((2.0000000000000142 2.0000000000000142, 6 2.0000000000000142, 6 3.6666666666666652, 2.0000000000000142 2.333333333333341, 2.0000000000000142 2.0000000000000142))

Table A.5: Table test_regions in Microsoft SQL Server after inserting both geometries and their intersection.

A_NAME	B_NAME	A_CONTAINS_B
region1	region1	TRUE
region1	region2	FALSE
region1	1intersection2	TRUE * (Right answer, but just by chance)
region2	region1	FALSE
region2	region2	TRUE
region2	1intersection2	FALSE * (Should have returned TRUE)
1intersection2	region1	FALSE
1intersection2	region2	FALSE
1intersection2	1intersection2	TRUE

Table A.6: Microsoft SQL Server answers to the contains test.

A.3.1 SQL Server commands

```
-- 1 - Table creation:
-- ----------------------
CREATE TABLE test_regions (
   my_id integer PRIMARY KEY,
   my_name VARCHAR(64),
   my_shape GEOMETRY
   );

-- 2 - Geometries insertion:
-- ----------------------------
-- 2.a) A rectangle:
INSERT INTO test_regions (my_id, my_name, my_shape) VALUES (
   1,
   'region1',
   GEOMETRY::STGeomFromText('POLYGON((2 1, 6 1, 6 5, 2 5, 2 1))', 0));

-- 2.b) A triangle:
INSERT INTO test_regions (my_id, my_name, my_shape) VALUES (
   2,
   'region2',
   GEOMETRY::STGeomFromText('POLYGON((1 2, 7 2, 7 4, 1 2))', 0));

-- 3 - Intersection computation:
-- --------------------------------
-- NOTE: Boundaries intersect at points (2, 2), (6, 2), (6, 11/3) and (2, 7/3).
INSERT INTO test_regions
   SELECT '12', '1intersection2', a.my_shape.STIntersection(b.my_shape)
   FROM test_regions a, test_regions b
   WHERE a.my_name = 'region1' AND b.my_name = 'region2';

-- 4 RESULTS:
-- -------------
-- 4.a) Table content
SELECT my_id, my_name, ST_ASText(my_shape) FROM test_regions;
```

```
-- MY_ID  MY_NAME          ST_ASText(MY_SHAPE)
-- ------ ---------------- ------------------------------------------------------------
-- 1      region1          POLYGON ((2 1, 6 1, 6 5, 2 5, 2 1))
-- 2      region2          POLYGON ((1 2, 7 2, 7 4, 1 2))
-- 12     1intersection2   POLYGON ((2.0000000000000142 2.0000000000000142,
--                                   6 2.0000000000000142,
--                                   6 3.6666666666666652,
--                                   2.0000000000000142 2.333333333333341,
--                                   2.0000000000000142 2.0000000000000142))

-- 4.b - Containment tests:
SELECT a.my_name AS a_name, b.my_name AS b_name,
  a.my_shape.STContains(b.my_shape) AS a_contains_b
  FROM test_regions a, test_regions b
  ORDER BY a.my_id, b.my_id;
-- A_NAME           B_NAME           A_CONTAINS_B
-- ---------------- ---------------- ------------
-- region1          region1          1
-- region1          region2          0
-- region1          1intersection2   1  * (Right answer, just by chance)
-- region2          region1          0
-- region2          region2          1
-- region2          1intersection2   0  * (* Should have returned 1)
-- 1intersection2   region1          0
-- 1intersection2   region2          0
-- 1intersection2   1intersection2   1
```

Appendix B

Publications and other research achievements

This appendix summarizes the publications and other research achievements of the author of this thesis. For some of the publication, we include references to relevant works in which they have been cited (these citations were updated in February 2012).

Research achievements directly related with this thesis

Publications in international journals

- J. A. Cotelo-Lema, L. Forlizzi, R H. Güting, E. Nardelli and M. Schneider. *Algorithms for Moving Objects Databases*. The Computer Journal, 46(6), June 2003, pages 680-712.

  According to Google Scholar this paper has been cited by 79 articles. Some of the most relevant ones are the following:

  - Victor Teixeira de Almeida, Ralf Hartmut Guting, and Thomas Behr. *Querying Moving Objects in SECONDO*. In Proceedings of the 7th International Conference on Mobile Data Management (MDM '06), Nara, Japan. IEEE Computer Society, Washington, DC, USA, pages 47-51. http://dx.doi.org/10.1109/MDM.2006.133
  - Nikos Pelekis, Babis Theodoulidis, Ioannis Kopanakis, and Yannis Theodoridis. 2004. *Literature review of spatio-temporal database models*.

The Knowledge Engineering Review (2004), 19, 3 (September 2004), pages 235-274. http://dx.doi.org/10.1017/S026988890400013X

– Glenn S. Iwerks, Hanan Samet, and Kenneth P. Smith. 2006. *Maintenance of K-nn and spatial join queries on continuously moving points*. ACM Trans. Database Syst. 31, 2 (June 2006), pages 485-536. http://doi.acm.org/10.1145/1138394.1138396

– Nikos Pelekis and Yannis Theodoridis. 2006. *Boosting location-based services with a moving object database engine*. In Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access (MobiDE '06). ACM, New York, NY, USA, pages 3-10. http://doi.acm.org/10.1145/1140104.1140108

– N. Pelekis, Y. Theodoridis, S. Vosinakis, and T. Panayiotopoulos. *Hermes – a framework for location-based data management*. In 11th International Conference on Extending Database Technology (EDBT'06), pages 1130-1134, 2006.

– Jeong, S.-H., Paton, N. W., Fernandes, A. A. A. and Griffiths, T. (2005), *An Experimental Performance Evaluation of Spatio-Temporal Join Strategies*. Transactions in GIS, 9: pages 129-156. http://dx.doi.org/10.1111/j.1467-9671.2005.00210.x

– Hui Ding, Goce Trajcevski and Peter Scheuermann. *Efficient Maintenance of Continuous Queries for Trajectories*. GeoInformatica Journal, Volume 12, Number 3, pages 255-288. http://www.springerlink.com/content/tv24k2041255p932/.

– Jose Macedo, Christelle Vangenot, Walied Othman, Nikos Pelekis, Elias Frentzos, Bart Kuijpers, Irene Ntoutsi, Stefano Spaccapietra, and Yannis Theodoridis, *Trajectory Data Models*. Book chapter in Mobility, data mining, and Privacy – Geographic Discovery, Fosca Giannoti, Dino Pedreschi (Eds), Springer 2008.

– Riccardo Ortale, Ettore Ritacco, Nikos Pelekis, Roberto Trasarti, G. Costa, F. Giannotti, G. Manco, C. Renso, and Y. Theodoridis. 2008. *The DAEDALUS framework: progressive querying and mining of movement data*. In Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems (GIS '08). ACM, New York, NY, USA, pages 411-415. http://doi.acm.org/10.1145/1463434.1463497

- J. A. Cotelo-Lema, R. H. Güting. *Dual Grid: A New Approach for Robust Spatial Algebra Implementation*. Geoinformatica Journal. Volume 6, Number 1, March 2002, pages 57-76.

  According to Google Scholar this paper has been cited by 15 articles. Some of the most relevant ones are the following:

  - Anthony J. Roy and John G. Stell. 2002. A Qualitative Account of Discrete Space. In Proceedings of the Second International Conference on Geographic Information Science (GIScience '02), September 25-28, 2002 . Max J. Egenhofer and David M. Mark (Eds.). Springer-Verlag, London, UK, pages 276-290.
  - Brian E. Weinrich and Markus Schneider. 2005. Use of rational numbers in the design of robust geometric primitives for three-dimensional spatial database systems. In Proceedings of the 13th annual ACM international workshop on Geographic information systems (GIS '05), November 04-05, 2005, Bremen, Germany. ACM, New York, NY, USA, pages 163-172. http://doi.acm.org/10.1145/1097064.1097088
  - Rod Thompson. *Towards a Rigorous Logic for Spatial Data Representation*. PhD Thesis, 2007, Publications on Geodesie 65 (ISSN 0165-1706) Delft: NCG Nederlandse Commissie voor Geodesie ISBN 978 90 6132 303 7.
  - Rodney James Thompson and Peter Oosterom. 2011. Connectivity in the regular polytope representation. Geoinformatica 15, 2 (April 2011), pages 223-246. http://dx.doi.org/10.1007/s10707-009-0094-3

Publications in international conferences

- [CL10] J. A. Cotelo-Lema, M. R. Luaces. *DualgridFF: a Robust, Consistent and Efficient Physical Data Model for Spatial Databases*. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '10), San José, California (Estados Unidos), 2010. ACM, New York, NY, USA, 422-425. http://doi.acm.org/10.1145/1869790.1869852.

- [Cot01] J. A. Cotelo-Lema. *An Analysis of Consistency Properties in Existing Spatial and Spatiotemporal Data Models*. Advances in Databases and Information Systems, 5th East - European Conference ADBIS' 2001, Vilnius (Lituania), 25th-28th September, 2001. Research Communications, A. Caplinskas, J. Eder (Eds.): Vol. 1(2001), pages 55-66.

Publications in national conferences

- J. A. Cotelo Lema. *Representación y Consulta Consistente de Información Espacial y Espaciotemporal*. In I Jornada de Sistemas de Información Geográfica (JSIG), Almagro (Ciudad Real), España. 19th November, 2001.

Awards

- The paper [CFG+03] was runner-up paper for the 2003 *The Computer Journal Wilkes Award*.
- The paper [Cot01] was awarded the *Best Student Paper Award* at the *Fifth East-European Conference on Advances in Databases and Information Systems (ADBIS'01)*.

Research stays

- August 1st, 1998 - July 31st, 2000. Hired as research employee (Wissenschaftlicher Angestellter) at the FernUniversität Hagen (Germany), under the supervision of Prof. Ralf Hartmut Güting. Hired under the EU funded project CHOROCHRONOS (a research network for Spatiotemporal Database Systems), involving 11 research groups from 8 different European countries.

  During the stay the Dualgrid approach was first conceptualized and the paper [CG02] was written and submitted. Also, preliminary works on paper [CFG+03] were done.

- September 3rd, 2001 - September 21st, 2001. Research stay at FernUniversität Hagen (Germany), under the supervision of Prof. Ralf Hartmut Güting.

  The stay was devoted to the final works on paper [CFG+03].

Other research achievements

Book chapters

- L. Montserrat, J. A. Cotelo, M. R. Luaces, D. Seco. *Collecting, Analyzing, and Publishing Massive Data about the Hypertrophic Cardiomyopathy*. Communications in Computer and Information Science - Biomedical Engineering Systems and Technologies, 52, Springer, Alemania, 2010, pp. 301-313. `http://dx.doi.org/10.1007/978-3-642-11721-3_23`

- N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, and J.R. Viqueira. *Sistemas de Información Geográfica: Revisión de su Estado Actual*. N.R. Brisaboa, ed., Ingeniería del Software en la Década del 2000, pp. 77-94. A Coruña, Spain, 2003.

Publications in International Journals

- N.R. Brisaboa, J. A. Cotelo, A. Fariña, M.R. Luaces, J.R. Paramá, J.R. Viqueira *Collecting and publishing massive geographic data*. Software - Practice and Experience 37(12), pp. 1319-1348, John Wiley & Sons 2007.

Publications in international conferences

- Montserrat, L.; Cotelo, J. A. ; Luaces, M. R.; Seco, *A Document Management System and Workflow to help at the Diagnosis of Hypertrophic Cardiomyopathy*. 2nd International Conference on Health Informatics (HEALTHINF 2009). INSTICC, Porto (Portugal), 2009.

- N.R. Brisaboa, J.A. Cotelo Lema, A. Fariñaa Martínez, M.R. Luaces, and J.R. Viqueira. *The E.I.E.L. Project: An Experience of GIS Development*. 9th EC-GI &GIS Workshop (ECGIS), A Coruña, Spain, 2003.

Publications in national and Ibero-American conferences

- C. Amil, N.R. Brisaboa, Cotelo Lema J.A, A. Fariña Martínez, M.R. Luaces, M.R. Penabad, A.S. Places, J.R. Viqueira. *Una Interfaz Web para un Sistema Geográfico de Información Turística*. II Jornada de Sistemas de Información Geográfica (JSIG). El Escorial, Spain, 2002.

- N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martinez, M.R. Luaces, J.R. Viqueira. *S.I.T.P.A.C.: A Territorial Information System for A Coruña Province*. 8th International Congress on Computer Science Research (CIICC). Colima, Mexico, 2001.

- N.R. Brisaboa, J.A. Cotelo Lema, M.R. Luaces, J.R. Viqueira. *State of the Art and Requirements in GIS*. 3rd Mexican International Conference on Computer Science (ENC). Aguascalientes, Mexico, 2001.

- N.R. Brisaboa, J.A. Cotelo Lema, A. Fariña Martinez, M.R. Luaces, J.R. Viqueira. *E.I.E.L.: Una Experiencia de un Desarrollo SIG*. I Jornada de

Sistemas de Información Geográfica (JSIG). Almagro (Ciudad Real), Spain, 19th November 2001.

Apéndice C

Descripción del trabajo presentado

C.1 Introducción

En las últimas décadas se ha dedicado un significativo esfuerzo a la integración de las tecnologías de Sistemas de Información Geográfica (SIG) con sistemas de información más tradicionales. Para dar soporte a esa integración, la tecnología de representación de datos espaciales ha sido mejorada en múltiples aspectos, desde los modelos (conceptuales y discretos) de representación de datos y lenguajes de consulta a las tecnologías de indexación y visualización y a los estándares de interoperabilidad. Como resultado de estos esfuerzos, las tecnologías SIG son ampliamente utilizadas en la actualidad en todo tipo de aplicaciones.

Las tecnologías y modelos espaciales utilizados por los sistemas SIG son clave para su evolución e implantación, razón por la cual la comunidad científica les ha dedicado importantes esfuerzos. Las tecnologías de bases de datos espaciales actuales ofrecen modelos de datos y operaciones estandarizados [OGC06], inspirados en *álgebras espaciales* con unas bases conceptuales sólidas. Para el estudio e implementación de modelos espaciales se distinguen típicamente tres niveles de abstracción: *modelos espaciales abstractos* (también denominados *modelos conceptuales*), *modelos espaciales conceptuales* (también denominados *modelos lógicos*), y *modelos espaciales físicos*.

Un *modelo espacial abstracto* se centra en describir los tipos de objetos geográficos del mundo real, las relaciones entre los diferentes tipos y las operaciones que

pueden ser realizadas sobre ellos usando conceptos formales definidos sin tener en cuenta aspectos de implementación. Un ejemplo de modelo espacial abstracto es el estándar internacional ISO 19107:2003 [ISO03], el cual define tipos conceptuales para representar objetos geográficos (como *curve* o *surface*) y operaciones sobre ellos (como *overlaps* o *touches*).

Un *modelo espacial discreto*, en cambio, tiene en cuenta las limitaciones de los sistemas informáticos (como son memoria física limitada o el rendimiento computacional) y define tipos de datos y algoritmos que pueden ser usados para implementar los conceptos de un modelo espacial abstracto. El estándar del Open Geospatial Consortium para *Simple Features in SQL* [OGC06] es un ejemplo típico de un modelo espacial discreto para el modelo abstracto definido por el estándar ISO 19107:2003. Este modelo define tipos de datos como *linestring* y *polygon*, teniendo en cuenta la necesidad de que la representación propuesta permita la implementación de algoritmos eficientes (con una adecuada complejidad algorítmica) para las operaciones que define.

Finalmente, un *modelo espacial físico* es una particular implementación de un modelo discreto en un determinado entorno informático. Por ejemplo, la implementación de OGC SFS en PostGIS [Ref10] para PostgreSQL usando GEOS define un modelo físico. Un modelo físico también define algunos aspectos dejados sin definir (abiertos) por el modelo discreto, como por ejemplo el espacio de representación de coordenadas a usar y la precisión usada en el mismo.

Los modelos espaciales abstractos definen un álgebra de tipos de datos espaciales y operaciones con propiedades teóricas sólidas. Por un lado, garantizan que estas operaciones son cerradas, es decir, que los tipos de datos propuestos son suficientemente potentes como para representar el resultado de cualquiera de las operaciones espaciales propuestas por el modelo (por ejemplo, un modelo espacial abstracto debe garantizar que el resultado de la operación *intersection* aplicada sobre dos valores cualesquiera del tipo *surface* puede ser representado usando un valor del tipo *surface*). Por otro lado, garantizan que el conjunto de operaciones propuestas cumple con unas ciertas propiedades lógicas (por ejemplo, que $A \cap B \subseteq A$).

Durante las últimas décadas se ha dedicado un gran esfuerzo a la definición de modelos espaciales abstractos, los cuales han alcanzado un grado de madurez suficiente para que incluso se hayan podido llegar a definir y aprobar estándares internacionales como el anteriormente mencionado ISO 19107:2003. Por contra, la definición de modelos espaciales discretos y físicos capaces de mantener las propiedades del modelo espacial abstracto ha demostrado ser un problema mucho más difícil de abordar.

Los modelos espaciales discretos han de definir tipos de datos usando un número finito de componentes. Por ejemplo, un modelo de datos discreto puede representar los objetos del tipo abstracto *curve* usando un conjunto de segmentos lineales, los objetos

del tipo abstracto *surface* usando un conjunto de segmentos lineales representando su contorno, y los objetos del tipo abstracto *point* con un par de coordenadas numéricas. Si el modelo espacial discreto continúa asumiendo el uso de un espacio de representación continuo, podrá probablemente afirmar que mantiene las propiedades del modelo abstracto, pero la resolución del problema de representar las coordenadas espaciales en un ordenador (la discretización del espacio geográfico) es postergada y trasladada a quien se encargue de la definición del modelo espacial físico. Si por el contrario se opta por afrontar el problema en el modelo espacial discreto, por ejemplo mediante el uso de un espacio finito para la representación de coordenadas (por ejemplo, enteros de 64 bits o números en coma flotante de doble precisión conforme al estándar IEEE754), entonces se vuelve realmente difícil lograr mantener las propiedades lógicas del modelo abstracto.

A pesar de los esfuerzos dedicados por la comunidad científica, las soluciones propuestas para abordar la discretización del espacio de representación distan de ser satisfactorias. En consecuencia, las implementaciones existentes en la actualidad de librerías y extensiones de bases de datos espaciales sufren severamente estas limitaciones. Aparentan cumplir con las álgebra conceptuales originales, pero en realidad incumplen la mayor parte de las propiedades en que están basadas esas álgebras. Más específicamente, los modelos físicos no mantienen sus propiedades de cierre bajo el conjunto de tipos de datos y operaciones implementados, y las soluciones aplicadas para solventarlo, normalmente algún tipo de *resultado aproximado*, no cumplen con las propiedades lógicas esperadas de las operaciones en cuestión. En consecuencia, el modelo físico resultante no es capaz de ofrecer una implementación consistente de las operaciones espaciales ofrecidas a los usuarios. Como resultado de todo esto, el desarrollo de aplicaciones basadas en las propiedades del modelo conceptual (por ejemplo, aplicaciones de análisis espacial) se vuelve mucho más difícil, si no imposible. De hecho, incluso la implementación del propio modelo físico se vuelve mucho más compleja, al no poder apoyarse ni siquiera en las bases teóricas del modelo conceptual que se supone se está implementando.

C.2 Metodología utilizada

El objetivo principal de esta tesis es sentar las bases para el desarrollo de extensiones de bases de datos espaciales capaces de cumplir las propiedades clave del álgebra espacial conceptual en la que se basan, teniendo en cuenta además las restricciones impuestas por la realidad de las aplicaciones GIS actuales en términos de rendimiento, de consumo de recursos y de interoperabilidad con las aplicaciones y estándares existentes.

La estrategia seguida para alcanzar este objetivo ha sido la siguiente:

- En primer lugar, analizamos el estado del arte actual en representación de información espacial, prestando especial atención a las limitaciones impuestas por los ordenadores y los efectos que esas soluciones tienen en el cumplimiento o incumplimiento de las propiedades del modelo conceptual. Igualmente estudiamos la capacidad de los modelos espaciales discretos y físicos existentes para trasladar a las aplicaciones del mundo real y a las extensiones de bases de datos espaciales toda la semántica y propiedades lógicas de los modelos conceptuales espaciales en los que se basan.

- En segundo lugar, identificamos cuales son los aspectos principales que el modelo espacial físico debe solucionar para garantizar que es capaz de mantener las propiedades clave de los modelos discreto y abstracto en los que se basa.

- En tercer lugar, proponemos un marco teórico para el diseño de modelos físicos (Dualgrid) que garantiza que las implementaciones de álgebras espaciales basadas en él mantienen las propiedades clave desde el punto de vista de las aplicaciones de usuario. Como pruebas de concepto, se adaptaron dos implementaciones existentes al uso de Dualgrid, basadas originalmente en planteamientos distintos a la hora de mitigar los problemas de discretización y siendo una de ellas una extensión de bases de datos espaciales ampliamente usada (PostgreSQL/PostGIS). Los resultados experimentales de dichas adaptaciones evidencian cómo el uso de Dualgrid soluciona los problemas de consistencia y (incluso) de implementación en ambas soluciones.

- Por último y en cuarto lugar, estudiamos de nuevo la propuesta de Dualgrid y la redefinimos para extender sus propiedades (DualgridFF) con el fin de hacer posible el cumplimento de las restricciones adicionales (en términos de rendimiento, espacio de almacenamiento e interoperabilidad) impuestas por las aplicaciones, tecnologías GIS y estándares de interoperabilidad (OGC) existentes.

Los resultados del análisis realizado en el primer y segundo paso han permitido establecer una estrategia novedosa en el planteamiento seguido para la resolución de los problemas de discretización del espacio de representación. Como resultado, las soluciones propuestas en el tercer y cuarto paso se caracterizan por el énfasis en mantener el cierre del espacio de representación usado, en vez de empeñarse en emular las propiedades que un espacio de representación pierde como resultado de su discretización. Gracias a este enfoque, Dualgrid y DualgridFF logran garantizar que el

modelo espacial físico resultante permanezca cerrado para las operaciones espaciales típicas, y muy especialmente para las relacionadas con teoría de conjuntos.

C.3 Conclusiones y contribuciones

Las tecnologías de bases de datos han alcanzado un elevado nivel de madurez en las últimas décadas. Actualmente, todos los sistemas gestores de bases de datos comerciales relevantes (PostgreSQL, MySQL, Oracle, DB2, Microsoft SQL Server, Informix, etc.) ofrecen tipos de datos y operaciones espaciales, normalmente implementando estándares ampliamente aceptados (SFS, ISO SQL/MM, etc.). Sin embargo, todas las implementaciones existentes sufren un problema fundamental: no son robustas/consistentes, en el sentido de que son incapaces de cumplir ni siquiera con las propiedades teóricas más básicas de los modelos abstractos que se supone implementan.

El problema de definir e implementar un modelo espacial físico que mantenga las propiedades definidas por un modelo espacial abstracto es un problema todavía no resuelto. El Apéndice A muestra un ejemplo muy sencillo que produce respuestas inconsistentes en tres de los principales SGBD espaciales comerciales, concretamente Oracle Spatial, PostgreSQL/PostGIS y Microsoft SQL Server. En algunos casos, estos problemas logran incluso causar errores fatales (*crashes*) y excepciones en sus algoritmos geométricos (como se pone en evidencia en la Sección 4.4 de esta tesis, con tests realizados sobre PostgreSQL/PostGIS, y como puede verse en la Sección 1.5.5 de la *Oracle Spatial Developer's Guide* [Ora10] con respecto al uso de tolerancias en Oracle Spatial). Como resultado, los desarrolladores de aplicaciones SIG y los usuarios de las mismas se ven obligados a soportar el inconveniente de gestionar las consecuencias de usar un álgebra espacial *rota*[1].

Aunque en la literatura científica hay propuestas que solucionan este problema al nivel del modelo espacial discreto [GS93], éstas no solucionan completamente el problema, y sus inconvenientes (tal y como se explica en la Sección 2.4.2) imposibilitan su uso como una solución para el desarrollo de tecnologías de bases de datos espaciales comerciales.

Este trabajo de investigación se ha centrado en cubrir el hueco existente entre las propuestas existentes en modelos espaciales discretos y la implementación de los

[1] Usamos aquí el término *álgebra rota* con el fin de remarcar el hecho de que el uso de una implementación que ha dejado de cumplir las propiedades del modelo abstracto arruina todos los esfuerzos dedicados en su momento a la cuidadosa definición del mismo, porque los usuarios ya no pueden apoyarse en las propiedades lógicas y matematicas que dicho modelo abstracto ofrecía.

mismos en SGBD espaciales y herramientas GIS. Como resultado, esta tesis aporta tres contribuciones principales:

- Realiza un estudio detallado del estado del arte y un análisis de los requisitos clave que un modelos espacial físico debería cumplir para garantizar que se mantiene cerrado ante las principales operaciones espaciales del modelo abstracto (en especial las correspondientes a la teoría de conjuntos), manteniendo así sus propiedades teóricas.

- La propuesta de un nuevo espacio de representación (Dualgrid) que cumple los requisitos mencionados y, en consecuencia, garantiza que se mantienen las propiedades del correspondiente modelo espacial abstracto, permitiendo la implementación exacta de sus operaciones (en lugar de aproximaciones de las mismas) y al mismo tiempo usar representaciones basadas en coordenadas de tamaño fijo. El uso de Dualgrid ha permitido reincorporar a dos implementaciones existentes (una implementación del álgebra ROSE y una versión de PostGIS/GEOS) aquellas propiedades de robustez y consistencia originalmente perdidas durante la implementación del modelo discreto.

- La propuesta del modelo espacial físico DualgridFF, una evolución de Dualgrid diseñada para cumplir los requisitos adicionales en términos de rendimiento e interoperabilidad de los usuarios profesionales[2], y orientada a su uso en nuevas implementaciones. DualgridFF ofrece todas las ventajas ya aportadas por Dualgrid, mejorando éste al permitir la implementación de modelos espaciales de calidad comercial, entendiendo ésta no sólo en términos de robustez y consistencia, sino también en términos de rendimiento, coste de almacenamiento e interoperabilidad.

Dualgrid y DualgridFF proveen una base sólida para resolver los problemas de robustez y consistencia de las tecnologías espaciales vectoriales actuales, atacando directamente a la raíz de dichos problemas. Ambos sirven de herramienta para permitir que los modelos espaciales físicos puedan mantenerse cerrados bajo las operaciones definidas por los modelos abstractos. Además, DualgridFF va un paso más allá, teniendo en cuenta las necesidades de todas las partes involucradas en el desarrollo y uso de tecnologías espaciales, de modo que las implementaciones de tecnologías de bases de datos espaciales basadas en él puedan conciliar con éxito las necesidades de las diferentes partes involucradas, ayudando a aunar esfuerzos en la evolución de los sistemas de información geográfica.

[2] Usamos aquí el término *usuarios profesionales* en contraposicion a los *investigadores académicos*.

Como resultado, las contribuciones de esta tesis permiten una mejora cualitativa de las tecnologías (y extensiones de bases de datos) espaciales existentes, lo que permitirá dotar a éstas de las sólidas bases formales que los desarrolladores necesitan para poder desarrollar aplicaciones SIG avanzadas.

C.4 Trabajo futuro

En este trabajo de investigación se ha dado un gran paso adelante para aportar al desarrollo de tecnologías de bases de datos espaciales las bases necesarias para implementar modelos de datos espaciales robustos y consistentes. A raíz de los resultados obtenidos, se identifican las siguientes posibles líneas de investigación futuras:

- *Mejora de librerías de gestión de información espacial existentes*. La manera más rápida y directa de trasladar las ventajas de las propuestas de este trabajo de investigación al mundo real de las aplicaciones SIG consiste en adaptar las librerías de gestión de información espacial existentes más ampliamente usadas, de modo que hagan uso de DualgridFF. Un claro candidato es la librería JTS Topology Suite (`http://sourceforge.net/projects/jts-topo-suite/`), una librería ampliamente utilizada por los entornos de desarrollo SIG, herramientas y aplicaciones para ofrecer capacidades de análisis espacial. Ofrecer una versión de JTS sobre DualgridFF (añadiendo una capa de precisión DualgridFF a las capas de precisión actualmente disponibles en JTS) permitiría a los desarrolladores de aplicaciones SIG simplificar sus implementaciones de análisis espacial y evolucionarlas a niveles de análisis más elevados, dado que el uso del modelo físico de DualgridFF permitiría que sus algoritmos de análisis espacial pudiesen contar con operaciones espaciales robustas y consistentes. Además, esa adaptación de JTS permitiría realizar una validación experimental de las estimaciones de rendimiento y necesidades de almacenamiento asociadas al uso de DualgridFF y de las técnicas de optimización propuestas en [CL10]. Al mismo tiempo, permitiría verificar la efectividad y facilidad de uso de DualgridFF para la adaptación de implementaciones existentes.

- *Mejora de extensiones de bases de datos espaciales*. Un segundo (pero no menos relevante) camino sería la mejora de extensiones de bases de datos espaciales existentes para adaptarlas al uso del modelo físico de DualgridFF. Esto permitiría a dichas extensiones espaciales ofrecer operaciones consistentes y, no menos importante, liberar a las mismas de las complejidades algorítmicas que habitualmente necesitan implementar para (intentar) minimizar

los problemas de robustez que sus (actuales) modelos discretos no cerrados generan. Como resultado, su robustez mejoraría y la complejidad interna de sus implementaciones se simplificaría, haciendo más fácil la incorporación futura de operaciones espaciales más potentes. Adicionalmente, esto nos permitiría analizar en más profundidad posibles estrategias de implementación para optimizar los requisitos de rendimiento y de espacio de almacenamiento, con el fin de reducir el impacto de incorporar DualgridFF a implementaciones existentes. Un candidato a dicha evolución es PostGIS/GEOS, la extensión de bases de datos espaciales de código abierto usada por PostgreSQL.

- *Optimización de librerías de precisión arbitraria a los requisitos específicos de DualgridFF*. Otra posible línea de investigación futura consiste en el análisis de algoritmos para la mejora del rendimiento de computaciones en precisión arbitraria en los casos específicos requeridos en DualgridFF. Dado que dichas computaciones en precisión arbitraria son requeridas en escenarios muy específicos, es muy probable que una librería específicamente diseñada pudiese exhibir un rendimiento mucho mejor que librerías estándar al ser usada en implementaciones de DualgridFF.

- *Desarrollo de funcionalidades de análisis espacial más avanzadas en entornos de desarrollo SIG existentes*. Nuestro grupo de investigación tiene una amplia experiencia en el desarrollo de herramientas SIG avanzadas. El desarrollo de soporte de análisis espacial en ellas se ha visto siempre dificultado por las limitaciones impuestas por las tecnologías espaciales actualmente disponibles, caracterizadas por su inconsistencia entre operaciones. La disponibilidad de librerías espaciales basadas en DualgridFF solventaría dichas limitaciones, abriendo la oportunidad de una evolución en la calidad y potencia del soporte de análisis espacial ofrecido por las aplicaciones SIG.

- *Extensión a modelos espaciotemporales*. Esta tesis ha estado centrada en modelos espaciales no temporales. Sin embargo, la última década se ha caracterizado por un importante esfuerzo investigador orientado al desarrollo de modelos de datos espaciotemporales y sus algoritmos [CFG+03], donde la dimensión temporal de la información espacial es tenida en consideración. Dependiendo del modelo, la evolución temporal de los datos espaciales puede ser discreta (el valor espacial cambia en instantes temporales específicos) o continua (donde los valores espaciales evolucionan de un modo continuo a lo largo del tiempo, habitualmente denominados en la literatura *moving objects*).

  El uso de Dualgrid y DualgridFF para la implementación de modelos espaciotemporales discretos debería ser relativamente sencillo. La implemen-

tación parcial realizada en [CFG+03] estaba preparada para el uso de datos espaciales siguiendo las restricciones de Dualgrid. Usando *static slices* (donde el valor espacial representado por el *slice* es estático durante todo el intervalo temporal) y cumpliendo las restricciones de Dualgrid, la implementación realizada era capaz de ofrecer respuestas consistentes entre operaciones espaciales y espaciotemporales.

La aplicación de Dualgrid y DualgridFF a *moving objects*, sin embargo, requiere un estudio más profundo. Debería ser posible extender Dualgrid a geometrías 3D (donde la tercera dimensión sería usada para el eje temporal), y representar la evolución de un objeto para cada *time slice* como una de esas geometrías 3D. De hecho, en [Tho07] los autores definen una representación 3D (en este caso usada realmente como 3D, con la altura como la tercera dimensión) basada en las lecciones aprendidas de Dualgrid. Esto ayudaría a mantener la consistencia entre operaciones espaciotemporales en *moving objects*. Sin embargo, la parte difícil estaría en lograr ofrecer consistencia entre operaciones espaciales y espaciotemporales. La razón es que los modelos espaciotemporales necesitan ofrecer soporte para la realización de proyecciones en la dimensión temporal (para, por ejemplo, poder obtener el instante en el cual un objeto espaciotemporal ocupó una determinada posición, o el intervalo de tiempo durante el cual el objeto ha existido) y en la dimensión espacial (para, por ejemplo, recuperar el valor espacial de un objeto espaciotemporal en un determinado instante, o la proyección de la trayectoria de todo el objeto espaciotemporal en el espacio). Por tanto, cualquier evolución de Dualgrid al dominio espaciotemporal (llamémosle DualgridST) tendría que definir como mínimo un espacio de representación tal que las proyección en el espacio de valores espaciotemporales fuese una operación cerrada. Adicionalmente, idealmente debería de ser capaz de soportar la combinación de proyecciones en el espacio con proyecciones en el tiempo. Si estas combinaciones no son gestionadas adecuadamente, secuencias de estas operaciones generarían muy probablemente valores espaciales que no serían representables en DualgridST. Esto es debido a que la combinación de proyecciones en el espacio y en el tiempo puede aumentar con cada proyección la precisión requerida para representar los valores temporales y espaciales. Este es un problema que debería ser cuidadosamente analizado y gestionado si se quiere extender Dualgrid a modelos espaciotemporales.

Bibliografía

[ABD+97] F. Avnaim, J-D. Boissonnat, O. Devillers, F. Preparata and M. Yvinec. *Evaluating signs of determinants using single-precision arithmetic*. In Algorithmica, Vol. 17, pages 111-132, 1997.

[BF09] R. Bulbul and A. U. Frank. *Big Integers for GIS: Testing the Viability of Arbitrary Precision Arithmetic for GIS Geometry*. Poster presented in 12th AGILE International Conference on Geographic Information Science Hannover, Germany, June 2-5, 2009.

[BM98] P. Burrough y R. McDonnell. *Principles of Geographical Information Systems*. Oxford University Press, 1998. ISBN: 0-19-823365-5.

[BO79] J. Bentley, T. Ottmann. *Algorithms for Reporting and Counting Geometric Intersections*. IEEE Transactions on Computers, C-28:643– 647, 1979.

[CG02] J. A. Cotelo-Lema, R. H. Güting. *Dual Grid: A New Approach for Robust Spatial Algebra Implementation*. Geoinformatica Journal. Volume 6, Number 1, March 2002, pages 57-76.

[Cha94] E. Chan, R. Zhu. *QL/G - A Query Language for Geometric Databases*. TR CS-94-25, Univ. of Waterloo, 1994.

[CFG+03] J. A. Cotelo-Lema, L. Forlizzi, R H. Güting, E. Nardelli and M. Schneider. *Algorithms for Moving Objects Databases*. The Computer Journal, 46(6), June 2003, pages 680-712.

[CL10] J. A. Cotelo-Lema, M. R. Luaces. *DualgridFF: a Robust, Consistent and Efficient Physical Data Model for Spatial Databases*. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '10), San José, CA (Estados Unidos), 2010.

ACM, New York, NY, USA, 422-425. http://doi.acm.org/10.1145/1869790.1869852.

[Cot01] J. A. Cotelo-Lema. *An Analysis of Consistency Properties in Existing Spatial and Spatiotemporal Data Models*. Advances in Databases and Information Systems, 5th East - European Conference ADBIS' 2001, Vilnius (Lituania), 25th-28th September 2001. Research Communications, A. Caplinskas, J. Eder (Eds.): Vol. 1(2001), pages 55-66.

[Cre06] R. Creo-Hombre. *Adaptación de GEOS y PostGIS a la tecnología Dualgrid*. TI 455, Universidade da Coruña (Spain), 2006.

[Dav98] J. Davis. *IBM's DB2 Spatial Extender: Managing Geo-Spatial Information Within The DBMS*. Technical report, IBM Corporation, 1998.

[DS90] D. Dobkin, D. Silver. *Applied Computational Geometry: Towards Robust Solutions of Basic Problems*. Journal of Computer and System Sciences, 40:70–87, 1990.

[Ege94] M. Egenhofer. *Spatial SQL: a Query and Presentation Language*. In IEE Transactions on Knowledge and Data Engineering. Vol 6, N°1, 1994, 86-95.

[FDP+99] A. Fernandes, A. Dinn, N. Paton, M. Williams, O. Liew. *Extending a Deductive Object-Oriented Database System with Spatial Data Handling Facilities*. Information and Software Technology, 41(1):483–497, 1999.

[For85] A. Forrest. *Computational Geometry in Practice*. En Fundamental Algorithms for Computer Graphics, pp. 707–723. Springer-Verlag, 1985.

[GdRS95] R. H. Güting, T. de Ridder, M. Schneider. *Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types*. En Proc. of the 4th Intl. Symposium on Large Spatial Databases, pp. 216–239, Portland, Maine, 1995.

[GK97] S. Grumbach, G. Kuper. *Tractable Recursion over Geometric Data*. In International Conference on Constraint Programming, 1997.

[GM95] L. Guibas, D. Marimont. *Rounding Arrangements Dynamically*. En 11th Annual Symposium on Computational Geometry, pp. 190–199, 1995.

[GRS98] S. Grumbach, P. Rigaux, L. Segoufin. *The Dedale System for Complex Spatial Queries*. En Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 213–224, 1998.

[GRSS97] S. Grumbach, P. Rigaux, M. Sholl, L. Segoufin. *Dedale: A spatial constraint database*. In Intl. Workshop in Database Programming Languages (DBPL'97), 1997.

[GS93] R. H. Güting, M. Schneider. *Realms: A Foundation for Spatial Data Types in Database Systems*. In Proc. of the 3rd. Intl. Symposium on Large Spatial Databases, pages 14-35. Singapore, June 93.

[GS95] R. H. Güting, M. Schneider. *Realm-Based Spatial Data Types: The ROSE Algebra*. In VLDB Journal. Vol. 4(2), pages 100-143, 1995.

[Gun88] O. Günter. *Efficient Structures for Geometric Data Management*. In Lecture Notes in Computer Sciences LNCS 377, Springer-Verlag. 1988.

[Güt88] R. H. Güting. *Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems*. En Proc. of the Intl. Conf. on Extending Database Technology, pp. 506–527, Venice, Italy, 1988.

[Güt89] R. H. Güting. *Gral: An Extensible Relational Database System for Geometric Aplications*. En In Proceedings of the Fifteenth International Conference on Very Large Data Bases, pp. 33–44, Amsterdam, 1989.

[Güt94a] R. H. Güting. *GraphDB: A Data Model and Query Language for Graphs in Databases*. Proc. of the 20th Intl. Conference on Very Large Databases, Santiago, 1994, 297 - 308.

[Güt94b] R. H. Güting. *Special Issue on Spatial Database Systems: An Introduction to Spatial Database Systems*. VLDB Journal: Very Large Data Bases, 3(4) (1994) 357-399.

[GY86] D. Greene, F. Yao. *Finite-Resolution Computational Geometry*. En Proc. 27th IEEE Symposium on Foundations of Computer Science, pp. 143–152, 1986.

[IEEE08] Institute of Electrical and Electronics Engineers (IEEE). *IEEE 754-2008 Standard for Floating-Point Arithmetic*. 2008.

[Ill94] *Illustra 2D Spatial Datablade (Release 1.3) Guide*, 1994.

[ISO03] ISO/IEC. *Geographic Information - Spatial Schema*. International standard ISO 19107:2003, 2003.

[ISO05a] ISO/IEC. *Geographic information - Rules for application schema*. International standard ISO 19109:2005, 2005.

[ISO05b] ISO/IEC. *Geographic information - Schema for coverage geometry and functions*. International standard ISO 19123:2005, 2005.

[ISO07] ISO/IEC. *Geographic information - Core profile of the spatial schema*. International Standard ISO 19137:2007, 2007.

[ISO07b] ISO/IEC. *Geographic information - Geography Markup Language (GML)*. International Standard ISO 19136:2007, 2007.

[ISO10] ISO/IEC. *Geographic information - Web Feature Service*. International Standard ISO ISO 19142:2010, 2010.

[LT92] R. Laurini, D. Thompson. *Fundamentals of Spatial Information Systems*. In Academic Press. APIC Series 37, 1992.

[LTR99] Lorentzos, N. A., Tryfona, N., Rios Viqueira, J. R. *Relational Algebra for Spatial Data Management*. ISD'99. LNCS 1737, (1999) 192-208.

[KM83] P. Kornerup, D. Matula. *Finite Precision Rational Arithmetic: An Arithmetic Unit*. IEEE Transactions on Computers, C-28:378–388, 1983.

[KPV95] B. Kuijpers, J. Paredaens, J. Van den Bussche. *Lossless Representation of Topological Spatial Data*. In M. J. Egenhofer and J. R. Herring, editors, Advances in Spatial Databases, 4th. Int. Symp., SSD'95, pages 1-13, Springer, 1995.

[Mic09] *SQL Server Database Engine*. In *SQL Server 2008 Books Online* (November 2009). See *Types of Spatial Data* Section. Retrieved in April 2010 from `http://msdn.microsoft.com/en-us/library/bb964711.aspx`.

[Mil89] V. Milenkovic. *Double Precision Geometry: A General Technique for Calculating Line and Segment Intersections Using Rounded Arithmetic*. En 30th Annual Symposium on Foundations of Computer Science, pp. 500–505, 1989.

[MPF+96] V. Muller, N. Paton, A. Fernandes, A. Dinn, M. Williams. *Virtual Realms: An Efficient Implementation Strategy for Finite Resolution Spatial Data Types*. En Proc. of the 7th Intl. Symposium on Spatial Data Handling - SDH'96, volume 2, pp. 11B.1–11B.13, Delft, The Netherlands, 1996. 1991.

[Mys10] *MySQL 5.5 Reference Manual, Chapter 11.17 (Spatial Extensions)*. Revision 21559, July 2010. Retrieved in July 2010 from http://downloads.mysql.com/docs/refman-5.5-en.a4.pdf.

[OGC05] Open Geospatial Consortium Inc. *OpenGIS Web Processing Service (WPS) 1.0.0 Interface Standard*, 2005. http://www.opengeospatial.org/standards/wps.

[OGC06] Open Geospatial Consortium Inc. *OpenGIS Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option*. OpenGIS project document OGC 06-104r4, August 2006. http://www.opengeospatial.org/standards/sfs.

[OGC06b] Open Geospatial Consortium Inc. *OpenGIS Web Map Service (WMS) 1.3.0 Implementation Specification*, 2006. http://www.opengeospatial.org/standards/wms.

[OGC07] Open Geospatial Consortium Inc. *OpenGIS Geography Markup Language (GML) 3.2.1 Encoding Standard*, 2007. http://www.opengeospatial.org/standards/gml.

[OGC07b] Open Geospatial Consortium Inc. *OpenGIS Catalogue Service (CSW) 2.0.2 Implementation Specification*, 2007. http://www.opengeospatial.org/standards/cat.

[OGC09] Open Geospatial Consortium Inc. *OpenGIS Web Feature Service (WFS) 2.0 Interface Standard (also ISO 19142)*, 2009. http://www.opengeospatial.org/standards/wfs.

[OGC09b] Open Geospatial Consortium Inc. *Web Coverage Service (WCS) 2.0 Implementation Standard*, 2009. http://www.opengeospatial.org/standards/wcs.

[Ora10] Oracle. *Oracle Spatial Developer's Guide 11g Release 2 (11.2)*. Reference E11830-06, March 2010. Retrieved in April 2010 from http://www.oracle.com/pls/db112/to_pdf?pathname=appdev.112/e11830.pdf.

[Ora97] *Oracle8i Spatial Cartridge*, 1997.

[Ora99] *Oracle8i Spatial: Features Overview*, 1999.

[OTU87] T. Ottmann, G. Thiemt, C. Ullrich. *Numerical Stability of Geometric Algorithms*. En 3rd ACM Symposium on Computational Geometry, pp. 119–125, 1987.

[Par95] J. Paredaens. *Spatial Databases, the Final Frontier*. In G. Gottlob and M. Y. Vardi, editors, Proceedings of the 5th International Conference on Database Theory - ICDT'95. Lecture Notes in Computer Science LNCS 893, pages 14-32. Springer-Verlag, 1995.

[PS85] F. Preparata, M. Shamos. *Computational Geometry*. Springer-Verlag, Berlin, Heidelberg, New York, 1985.

[PVV94] J. Paredaens, J. Van den Bussche, D. Van Gucht. *Towards a Theory of Spatial Database Queries*. In Proc. 13th. ACM Symp. on Principles of Database Systems, pages 279-288, 1994.

[Ref10] Refractions Research. *PostGIS 1.5.1 Manual*. March 2010. Retrieved in April 2010 from `http://postgis.refractions.net/download/postgis-1.5.1.pdf`.

[Rig94] Ph. Rigaux, M. Scholl. *Multiple Representation Modelling and Querying*. In IGIS94, 1994.

[RSV01] P. Rigaux, M. Scholl, y A. Voisard. *Spatial Databases With Application To GIS*. Academic Press, 2001. ISBN: 1-55680-588-6.

[Sam90] H. Samet. *The Design and Analysis of Spatial Data Structures*. In Addison-Wesley, 1990.

[Sch94] P. Schorn. *Degeneracy in Geometric Computation and the Perturbation Approach*. The Computer Journal, 37(1), 1994.

[SH91] P. Svensson, Z. Huang. *Geo-Sal: A Query Language for Spatial Data Analysis*. En Proceedings of the 2nd Intl. Symposium on Large Spatial Databases, pp. 119–140, Zürich, Switzerland.

[Tho07] Rod Thompson. *Towards a Rigorous Logic for Spatial Data Representation*. PhD Thesis, 2007, Publications on Geodesie 65 (ISSN 0165-1706) Delft: NCG Nederlandse Commissie voor Geodesie ISBN 978 90 6132 303 7.

[Viv03] Vivid Solutions. *JTS Topology Suite Technical Specifications, Version 1.4*. October 2003. Retrieved in April 2010 from `http://www.vividsolutions.com/jts/bin/JTS%20Technical%20Specs.pdf`.

[Wor04] M.F. Worboys. *GIS: A Computing Perspective*. CRC, 2004. ISBN: 0415283752.

[WS05] B. E. Weinrich, M. Schneider . *Use of rational numbers in the design of robust geometric primitives for three-dimensional spatial database systems*. 13th ACM Int. Symp. on Advances in Geographic Information Systems (ACM GIS), 163-172, 2005.

[ZZC+02] Y. Zhang, L. Zhou, J. Chen and R. Zhao. *K-Order Neighbor: the Efficient Implementation Strategy for Restricting Cascaded Update in Realm*. Computational Science (ICCS 2002), Lecture Notes in Computer Science, 2002, Volume 2331/2002, 994-1003.