

# An Analysis of Consistency Properties in Existing Spatial and Spatiotemporal Data Models\*

José Antonio Coteló Lema

Laboratorio de Bases de Datos. Facultad de Informática. Universidade da Coruña.  
Campus de Elviña s/n. 15071. A Coruña. Spain.  
jaclema@mail2.udc.es  
jose-antonio.coteló-lema@fernuni-hagen.de

**Abstract** In the past, a big research effort has been spent in the development of database systems capable of efficiently storing, manipulating and retrieving spatial information. As a result, there have been important advances in the development of spatial and temporal data models and their integration (spatiotemporal models), query languages, indexing methods and user interfaces. Nowadays extensible database technology allows one to transfer the resulting technology into commercial databases, providing them with capabilities unknown just some years ago. However, due to the inherent difficulties of implementing such models on computers (where only discrete representations of spatial objects can be used), currently existing implementations tend to pay too little attention towards providing consistent implementations for the spatial operations offered to the users. As a result, the development of applications that depend on the correctness of these operations becomes much more complex, if not impossible.

In this paper we analyse existing spatial data models and proposed implementations with the aim of showing the strong and weak points of each approach with regard to consistency. We then study the consistency properties of possible extensions of those data models to the spatiotemporal domain. We show that the use of a carefully selected discrete space can considerably improve the consistency of the operations provided over it, offering to the developers a far more robust data model.

## 1 Introduction

In the past, a considerable research effort has been concentrated in the development of database systems capable of efficiently store, manipulate and retrieve spatial information. This effort has lead (is leading) to important advances in the development of spatial, temporal and spatiotemporal data models and query languages, indexing methods and user interfaces, providing database systems with a better support for the management of static and non static spatial data. Thanks to the existing extensible databases technology the transference of all this technology into commercial databases is becoming possible, providing them with capabilities unknown just some years ago.

One of the most challenging (and still open) problems in the design and implementation of spatial and spatiotemporal data models are those arising from the fact that at the abstract (theoretical) level spatial and spatiotemporal data models assume an underlying continuous space ( $R^n$ ), but at the implementation level the space has to be represented through discrete approximations (usually some kind of discrete grid), given the finiteness of computer resources. Unfortunately, such a “licence” turns out to have several and severe drawbacks such as robustness, topological correctness and consistency problems, given that most of the operations that were closed over the spatial data types in the continuous space are not closed any more when implemented over the (selected) discrete space. For instance, if the space is represented as a grid of points (the usual approach), it can happen that the intersection point of two lines does not belong to that grid and hence cannot be represented. Although several solutions can be applied to reduce those problems, they usually consist in implementing some kind of *approximated operations* instead of the original ones, or in restricting the set of operations provided to those that remain closed. However, the

---

\* This work has been partially supported through a predoctoral grant by the Universidade da Coruña.

former solution generates consistency problems, because those approximated operations do not fulfil the properties that the original operations were expected to fulfil, and the latter solution restricts severely the power of the original data model.

Although (in general) currently existing spatial and spatiotemporal data models and implementations pay attention to solve the robustness and topology problems arising from using a discrete space, very little attention is paid to providing consistency among operations as well. As a result, it is not unusual to find (commercial) implementations of spatial models where the intersection point of two lines can be a point that does not belong to any of the intersecting lines, or where a point belongs to two regions  $A$  and  $B$  but does not belong to their intersection. Although those inconsistencies can be acceptable in some application domains (e.g. spatial information visualisation), in other domains these inconsistencies can be a handicap for the development of applications, because if the possible inconsistencies are not carefully taken into account they can lead the application to reach unexpected states.

The goal of this article is to analyse the existing spatial data models (and proposed implementations) from the perspective of the consistency among operations<sup>1</sup>, with the aim of showing the strong and weak points of each approach with regard to consistency. An analysis of the possible consistency problems that an extension of those data models to the spatiotemporal domain could have is also performed. It will be shown that the use of a carefully selected discrete space can considerably reduce the resulting consistency problems, providing the developers with a far more robust data model.

This paper is structured as follows: Section 2 makes a review of the more common spatial representations. Section 3 explains the relevance of consistency problems in the development of spatial applications. Section 4 reviews several techniques used in spatial databases implementation to deal with inconsistencies. Section 5 compares the consistency properties of the representation models/approaches introduced in previous sections. Section 6 remarks the peculiarities of inconsistencies in environments where the time dimension is relevant and Sect. 7 analyses the consistency properties of all those models/approaches when extended to the spatiotemporal domain. Finally, Sect. 8 concludes the paper.

## 2 Representation of Spatial Data over Discrete Spaces

In the scientific literature several spatial data models have been proposed, some of which are nowadays widely used. They try to provide an efficient and finite representation for the usually infinite sets of points of spatial objects. These models can be roughly classified into four basic categories, depending on the approach followed in the representation and the intended target application domains. These four categories are (partially following [19]):

1. Raster models. Raster models are based in the concept of “bit-maps”. The infinite points of the space are represented by a finite number of raster points, which are uniformly distributed over the space. Raster representations have some advantages (i.e. set theory operations are easily implementable), but also important drawbacks, as the big storage requirements (all the points belonging to the object have to be explicitly represented).
2. Vectorial models. In the Spaghetti or Vectorial models, the information in the  $n$ -dimensional space is represented using  $m$ -dimensional hyperspaces, with  $m < n$ . Explained more informally, the infinite set of points belonging to a spatial object is represented by representing its boundary, normally using linear representations. For instance, in the two dimensional space one can represent points, graphs (whose data structure is a finite set of pairs of points), polylines (represented as a finite sequence of points), polygons (represented as a closed and no self-intersecting polyline) and 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).

---

<sup>1</sup> For the sake of clarity, in the rest of the paper we will call it just *consistency*.

One of the advantages of these models is the existence of efficient data structures [12], as well as very efficient algorithms for detecting properties on them [21].

This family of data models is widely used and numerous query languages [2,20,15,3] and algebras [13,14] are based on it.

3. Constraint databases models. Under the constraint databases model, data are represented using linear constraints. The same approach can be used for representing spatial data, as shown in [16,5,7,6]. For example, the constraint  $x \geq 1 \wedge x - y - 1 \geq 0 \wedge y \leq 7$  can represent 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, as all other data types.
4. Descriptive models. These models try to represent spatial objects through their qualitative properties, instead of through their quantitative (numeric) ones. Instead of representing the set of points belonging to each object stored in the database they just store their topological properties and the relationships between objects. The advantage of this approach is that, given that no quantitative data are stored, there are no problems arising from discretisation of the underlying space<sup>2</sup>. Therefore, it is mainly used in application domains where the qualitative analysis is the main aspect in the management of spatial data, as knowledge discovery or automated spatial reasoning.

This paper will focus its attention into vectorial data models, given that they are the ones where inconsistency problems are more relevant. Nevertheless, to put in context the analysis here performed, we will also compare the consistency properties of the different solutions used in vectorial models with the ones exhibited by the other three main data model types.

### 3 Understanding the Relevance of Consistency in Application Development

As explained previously, spatial data models are usually defined over an infinite space, usually  $R^n$ . However, when implementing them in computers such an infinite space must be replaced by some kind of finite approximation. For example, one can use a grid  $K^n$ , where each point coordinate belongs to  $\{-m, \dots, -1, 0, 1, \dots, m\}$ . As a result of such an approximation, however, the closure properties of the original (infinite) model are usually broken. For example, the intersection of two segments can yield as result a point that does not belong to  $K^n$ . There are two basic solutions to this problem: to provide only those operations that remain closed in the new 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. The second solution, in the other hand has several drawbacks. First of all, the use of approximated operations implies that they will not exhibit the properties that they are supposed to fulfil (i.e. their results will be inconsistent with their supposed properties) and, for example, it can happen that the intersection point of two segments does not belong to any of them. As a result, the implementation of spatial operations will not consist in a direct and (more or less) straightforward translation of geometry theory algorithms to a programming language, because such a direct translation would easily reach unexpected states and fail. Instead, their algorithms need to be redesigned to ensure that despite of these inconsistencies they will be robust (always finish) and return topologically correct results. Moreover, even if such goals are achieved, the inconsistencies among operation results are unavoidable.

Whether any of the two solutions described above is acceptable depends mainly on the application domain where the resulting implementation is going to be used. If the spatial data model is used just to graphically represent spatial objects, then inconsistencies are in general 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 from the theoretical one for the user's eyes. In those domains, it is usually more interesting to provide a powerful set of operations than to ensure

---

<sup>2</sup> In fact, there is no explicit underlying space.

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

Given that the set of operations that can be consistently provided together depends on the specific discrete space and spatial representation used, these two become a key aspect when selecting the spatial data model implementation to be used for one given application domain.

## 4 Approaches for Dealing with Inconsistencies in Spatial Data Models

Although the *approximation* approach can be acceptable for certain application domains (e.g. spatial information visualisation), there are some other domains (automated spatial reasoning, spatial knowledge discovery, etc.) where the consistency among certain operations is essential, requiring the use of some other solutions capable of reducing the inconsistencies and their consequences as much as possible. Among the different solutions proposed in the bibliography for improving the consistency of spatial models are the following:

- **Operations restriction:** The idea is that the algebra will only provide a set of operations that remains closed over the discrete implementation. With that purpose a subset of operations is removed from the original algebra 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 in the target domain.

The disadvantage of models following this approach is that they reduce drastically the set of available operations, and for example they do not provide set theory operations. The sets of operations that they provide are basically restricted to spatial predicates (e.g. topology relationships) and numerical operations (e.g. distance, area, etc.).

- **Orthogonal restriction:** If only spatial objects with boundaries composed of segments orthogonal to the plane axes are allowed, then the model remains closed under most of the spatial operations. For example, set theory operations are closed. An example of this approach is [17]. The advantages of this approach are its simplicity (both, the approach itself and the implementation of most of the operations over it) and that not even numerical problems need to be addressed in the algorithms implementation. The disadvantages are that it reduces the expressiveness of the spatial representation, and implies a high storage overhead for representing complex objects.

- **Exact representation:** With this approach the infinite space  $R^n$  is replaced by a  $Q^n$  space. Variable length rationals are used for representing spatial coordinates. The advantage of this approach is that (almost) all the geometric operations that are closed in  $R^n$  are also closed in  $Q^n$  (supposing linear representations are used), and therefore the source of inconsistency problems in the implementation of spatial models disappears. Although the coordinates of such an space are still (in theory) infinite numbers, the key issue is that the result of any of the operations provided in the algebra can be represented with coordinates of finite size, which will be proportional to the coordinate size of the original data and the number and type of operations applied to them. Therefore, the computer will be able to represent them.

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 the use of large numbers<sup>3</sup>). Also, the complexity of the required data structures increases: compact representations (records) are more efficient and usually preferred in databases implementations, but the use of

---

<sup>3</sup> Although the appropriated use of numeric filters can reduce it.

variable size coordinates makes necessary the use of more complex data structures. Therefore, this approach is sometimes used in geometric computation, but not in databases.

- **Realms approach:** In this approach, proposed in [10,11], all intersections between spatial objects stored in the database are made explicit at insertion/update time, modifying slightly 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[9,18]. As disadvantages, consistency is not guaranteed when the set of data stored in the databases changes (e.g. updates). Also, the implementation of transactions 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  $B$ . An example of an algebra implementation based on *realms* is the ROSE algebra[11].

- **Dual grid approach:** In this approach, a carefully selected grid of finite size rational numbers for point coordinates and a well selected set of segments are used as the pair of grids over which spatial objects are represented. The advantage of this approach is that the use of the two carefully defined grids allows the definition of spatial data types using fixed size coordinates in such a way that they are closed under a wide set of operations, including all the operations defined in the ROSE algebra[11]. It solves elegantly the consistency problems and the operations provided can be directly implemented using (without any modification) any of the existing algorithms available in the geometry literature. A formal definition of the dual grid approach and its properties can be found in [1].

## 5 Comparison of Consistency Support in the Spatial Dimension

To compare the consistency properties of the available solutions, it is interesting to identify the different sets of operations that are typical of several application domains. That way, given a domain and the set of operations over which consistency is relevant on it, it would be possible to analyse which data model families and/or consistency solutions (in the case of the vectorial model) are the more suitable ones for that domain.

We will classify the spatial operations provided by spatial algebras in the following classes<sup>4</sup> :

- Spatial predicates. This set of operations returns a boolean value depending on whether their spatial arguments fulfil some property. Examples of these operations are *equal*, *disjoint*, *inside*, *intersects*, *adjacent*, etc.
- Set theory operations. Operations implementing the set theory operations over spatial data, as *intersection*, *union* or *difference*.
- Decomposition operations. These operations decompose spatial objects in their components, more exactly in the components of their boundary. Examples of these operations are *vertices* and *contour*.
- Construction operations. They build spatial objects from their components. Examples are *interior* (returns the region enclosed by a *line* object), *segment* (returns a *line* object as the segment between two points) or *convexhull* (returns the smallest convex region such as the set of points given as argument are contained on it).
- Numeric operations. They return numeric results. Examples are *area*, *perimeter*, *length* (of a *line*) and *distance*.
- Scaling and rotation operations.
- Translation operations.

---

<sup>4</sup> For the sake of illustration some examples of operations belonging to each group are given, using (when possible) the names proposed in the ROSE algebra[11].

Spatial predicates can be consistently implemented in every family of spatial data models, given that they do not need to return spatial objects. Moreover, in the descriptive model they are trivial, given that the spatial data are already represented through their spatial relationships. The same happens with numeric operations, although in this case it is not possible to provide them in the descriptive model, given that no quantitative (numeric) information about the spatial objects is stored on that model.

The raster models can provide consistent implementations of set theory and translation operations. Scaling operations can also be provided, although in their non-consistent version. However, decomposition and construction operations have no meaning (and therefore cannot be implemented) here, given that objects have no boundaries in these models.

Constraint models can provide consistent versions of set theory operations. Similarly happens with decomposition operations, except for the *vertices* operation, which can only be provided as an approximated version<sup>5</sup>. With regard to construction operations, a consistent implementation of *interior* can be provided, but only an approximated one for *segment* and *convexhull*. Only non-consistent versions of scaling and rotation as well as translation operations can be provided.

In descriptive models only spatial predicates can be consistently implemented. All other types of operations require quantitative information and therefore cannot be provided.

With regard to vectorial models, the capabilities depend on the approach used. Models following the operations restriction approach can provide consistent implementations for all types of operations except for set theory and scaling and rotation operations, which cannot be provided. Translation operations can also be consistently provided, but only if the grid used is uniform (same distance among adjacent points, as for example when using integer coordinates). Otherwise they cannot be provided. The same happens with the approximation approach, although in this case non-consistent implementations of set theory and scaling and rotation operations can also be provided (as well as for translation operations in case the grid is not uniform). The orthogonal restriction approach behaves as the approximation approach, except that it allows the consistent implementation of set theory operations but only approximated implementations of construction operations (only *interior* can be consistently provided). The properties of both *realm* and *dual grid* based approaches are very similar, allowing the non-consistent implementation of scaling and rotating operations, translation operations and construction operations other than *interior*, and consistent implementations of *interior* and all the remaining types of operations. The main difference among the *realms* and *dual grid* based approaches are that the last one allows the non-consistent implementation of *convexhull* and that the *realms* only ensure consistency until the set of data stored in the database changes.

Finally, variable length rationals models can provide almost all of the operations consistently. The only exception are rotation operations, given that they can generate non-rational coordinates.

In Table 1 the consistency properties of the different data models for each class of operations is shown. Symbol *C* indicates that the set of operation can be provided with consistency. Symbol *A* (approximated) indicates that the operations can be provided, but without consistency. Symbol '–' indicates that the operations cannot be provided and/or have no sense on that model. Numbers in brackets have the following meanings:

1. *Vertices* can only be provided through an approximated implementation.
2. Only a consistent implementation of *interior* can be provided. *Realms* cannot provide *convexhull*.
3. Scaling can be provided consistently, whereas rotations cannot.
4. Consistency can be achieved only if an uniform grid is used.

It can be seen that except (of course) the variable length rationals approach, the models that provide a wider support for consistent implementation of spatial algebras are the constraint model and the vectorial model based on *realms* or *dual grid*, although their weak points are the construction and translation operations. Some of the remaining approaches can provide also a wide set of operations, but their capabilities to provide consistency are more limited.

<sup>5</sup> We are assuming that points are represented through their coordinates. If they are represented as the intersection of two lines then *vertices* could be consistently provided.

**Table 1.** Consistency properties of operations for each data model in the spatial domain.

	Raster models	Constraint models	Descriptive models	Vectorial models					
				Approximation	Operations restric.	Orthogonal restric.	Var. Length Rationals	Realms	Dual Grid
Spatial predicates	C	C	C	C	C	C	C	C	C
Set theory operations	C	C	-	A	-	C	C	C	C
Decomposition operations	-	C/A <sub>(1)</sub>	-	C	C	C	C	C	C
Construction operations	-	C/A <sub>(2)</sub>	-	C	C	C/A <sub>(2)</sub>	C	C/A <sub>-(2)</sub>	C/A <sub>(2)</sub>
Numeric operations	C	C	-	C	C	C	C	C	C
Scaling and rotation operations	A	A	-	A	-	A	C/A <sub>(3)</sub>	A	A
Translation operations	C	A	-	C/A <sub>(4)</sub>	C/- <sub>(4)</sub>	C/A <sub>(4)</sub>	C	A	A

## 6 Relevance of Time Dimension in Consistency

When extending the previous models to the time dimension (spatiotemporal models), there are two different time dimensions that can be considered, namely transaction time and valid time. The transaction time dimension refers to the time interval in which one (spatial) object has had a given value in the database. The valid time dimension refers to the time interval at which the object has had that value in the real world. Transaction time is always discrete, whereas valid time can be either discrete (if the object's value suffers only discrete changes) or continuous (if the spatial value can change continuously). One example of spatiotemporal objects with continuous valid time are moving objects [8,4].

In the case of spatiotemporal models, the consistency of projection operations (operations that perform some kind of projection of spatiotemporal values in the space or in the time dimension) has usually a huge relevance in (almost) any application domain. Some examples of such operations are, following the notation in [8], *deftime*, *locations*, *trajectory* or *atinstant* (the last one returns the spatial value of a spatiotemporal object at a time instant  $t$ ). Such a consistency of projection operations is more likely to be achieved in spatiotemporal models with discrete changes, whereas it becomes much more complex in continuous change data models (moving objects). The reason is that whichever representation is used, the possibility of recursively projecting spatiotemporal objects in the space and the time (i.e., generating a projection in the space and using the result to generate a projection in the time, and using such a result for generating another projection in the space, and so on) means that spatiotemporal coordinates can grow to an arbitrary length.

## 7 Comparison of Consistency in Spatiotemporal Models

One aspect to be considered when extending the previously shown data models/approaches to the spatiotemporal case is whether they can be extended at all and, if they can, whether they can provide consistency for the projection operations.

For the spatiotemporal domain we will not focus our attention into the same sets of operations than in the spatial case. The reason is that the spatiotemporal extension of those operations (what some authors call *lifted operations* [8]) usually has a uniform behaviour and, either all of them maintain their consistency properties or all of them lose them.

Therefore, what will be analysed in this section are the following sets of operations (examples are given based in [8]):

- Projection into space. Operations that project spatiotemporal objects into the space. Examples are *locations*, *trajectory* and *atinstant*.
- Projection into time. Operations that project spatiotemporal objects into the time dimension. An example is *deftime*.
- Lifted predicates. Similar to the spatial predicates, but they return a *moving(boolean)*, this is, a time-dependent value that is *true* for the time intervals for which the corresponding spatial predicate is *true*, and *false* for the time intervals for which it is *false*.
- Lifted operations. They are the spatiotemporal version of the spatial operations shown in Sect. 5. They return a value of the associated spatiotemporal type, which takes for each time instant the result of applying the corresponding spatial (non lifted) operator to the values that its arguments take at such time instant.
- Temporal restrictions. They restrict spatiotemporal values to some time interval. An example is the operation *atintervals*.
- Spatial restrictions. They restrict spatiotemporal values to some spatial area (e.g. a region). An example is the operation *at( $\alpha$ ,  $R$ )*, which restricts the spatiotemporal value  $\alpha$  to take values only in the area covered by the region  $R$ .

In the spatiotemporal context, apart from the consistency criteria described previously there is another important one: the result of a lifted operation must return the same value as if the non lifted operation were applied at each time instant to the arguments. For example, the lifted version of the *intersection* operation over regions takes as arguments two *moving(regions)* (spatiotemporal regions)  $A$  and  $B$  and should return, when consistently implemented, a *moving(region)*  $R$  such as

$$\forall t, \text{atinstant}(R, t) = \text{atinstant}(A, t) \text{intersection} \text{atinstant}(B, t) .$$

It is obvious that (almost) every of the previously described models can be extended to the spatiotemporal case with discrete changes (for both, transaction time and valid time) and still retain the consistency properties of the original (spatial) model. There is, however, an exception to this, namely the vectorial model over *realms*. In the case of transaction time, we have said previously that on this approach a spatial object  $A$  could be modified by the insertion of a second object  $B$ . Therefore, the consistency of operations cannot be ensured over the (transaction) time. In the case of valid time, the consistency properties of the original (spatial) version are preserved. However, given that when several spatial values are allocated in the same area of the plane the probability of needing to make explicit intersection points increases, and given that one should expect that the different values that a spatial objects takes over the time are allocated in the same area of the plane, the storage overhead of using a *realm* based representation could become much more relevant.

Of course, an extension of the descriptive model to spatiotemporal databases (discrete time) would not provide lifted spatial operations and spatial restrictions (quantitative information would be needed).

With regard to continuous valid time, it cannot be used at all with raster models or vectorial models with orthogonal restriction or operations restriction. The *realms* based approach cannot be extended either to the continuous valid time, because it would be needed to explicitly represent all the intersection points of all the possible spatial projections of the objects. Continuous time has also no sense in descriptive models, given that the spatial relationships among objects do not change continuously. Vectorial models with approximated approach can of course be used, but they will not provide any consistency for the operations described above, not even for lifted predicates, given that they will not fulfil the consistency restriction introduced previously in this section. Finally, the constraint databases model and vectorial models with *dual grid* could be extended to the spatiotemporal case and implement consistently lifted predicates and operations (with the limitations seen in the spatial case). Spatial and temporal restrictions could also be consistently implemented in both cases. However, for spatial and temporal projections the *dual grid* based models have a slight advantage over constraint databases models (at least for those using a general constraint databases approach), being able to provide consistency in one of the projections (either in time or in the space). The reason is that whereas (in principle) a constraint

databases spatiotemporal model would use the same precision for constraint coefficients for spatial and spatiotemporal values, a higher precision would be needed to be able to represent projections into the temporal or spatial dimensions. In a vectorial model, on the other hand, such a restriction does not exist, given that algorithms are implemented *ad-hoc* for each operation, in contrast with spatial constraint databases which take advantage of being able to implement spatial operations using the standard methods of constraint databases.

Variable length rational models can again provide consistent implementations for all operations (except for the rotation operations and associated lifted versions).

**Table 2.** Consistency properties of operations for each data model in the spatiotemporal domain.

Model/Approach	Discrete Transaction Time						Discrete Valid Time						Continuous Valid Time						
	Space projections	Time projections	Lifted predicates	Lifted operations	Spatial restrictions	Temporal restrictions	Space projections	Time projections	Lifted predicates	Lifted operations	Spatial restrictions	Temporal restrictions	Space projections	Time projections	Lifted predicates	Lifted operations	Spatial restrictions	Temporal restrictions	
Raster	C	C	C	C/A/-	C	C	C	C	C	C/A/-	C	C	-	-	-	-	-	-	
Constraint	C	C	C	C/A	C	C	C	C	C	C/A	C	C	A	A	C	C/A	C	C	
Descriptive	C	C	C	-	-	C	C	C	C	-	-	C	-	-	-	-	-	-	
Vectorial	Approximation	C	C	C	C/A	C	C	C	C	C	C/A	C	C	A	A	A	A	A	A
	Oper. Restr.	C	C	C	C/-	C	C	C	C	C	C/-	C	C	-	-	-	-	-	-
	Orthog. Restr.	C	C	C	C/A	C	C	C	C	C	C/A	C	C	-	-	-	-	-	-
	Var. Length Rat.	C	C	C	C/A	C	C	C	C	C	C/A	C	C	C	C	C	C/A	C	C
	Realms	A	A	A	A	A	C	C	C	C	C/A	C	C	-	-	-	-	-	-
Dual Grid	C	C	C	C/A	C	C	C	C	C	C/A	C	C	C/A(1)	C/A(1)	C	C/A	C	C	

Table 2 summarises the capabilities of each type of data models in the spatiotemporal domain. Symbols *C*, *A* and *-* have the same meaning as in Sect. 5. Wherever several levels of consistency are indicated (e.g. *C/A*) it means that for each specific operation of that type the consistency achieved is the same as in the spatial only model. The only exception are fields marked as (1). In those cases it means that only one of the projections (in the space or in the time) can be consistently provided, whereas the other one must be provided through an approximated implementation.

## 8 Conclusions

We have analysed in this paper 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 (specially prone to experience consistency problems), we have analysed the capabilities provided by the more common/promising approaches followed in their implementation.

As a result, it can be seen that the representation model/approach used restricts the sets of operations over which consistency can be ensured. Therefore, when implementing a spatial or spatiotemporal package addressed towards an specific set of application domains it is worthwhile to analyse the consistency requirements of those domains and take them into account when choosing the representation model to be used.

It can be seen in the previous sections that in the spatial domain constraint models and several approaches of vectorial models are close in their consistency capabilities, only beaten by the variable length rationals approach. In the spatiotemporal domain however, *realms* lose most of their capabilities, given their inability of providing consistency when transaction time has to be taken into account or to work with continuous changes. In contrast, *dual grid* keeps all its

properties under discrete time, and still keeps most of them with continuous changes, beating even the constraint models.

One last remark: in this paper only consistency issues have been taken into account. However, other aspects (i.e. performance or storage requirements) should also be considered when choosing the representation to be used. For example, variable length rationals representations can provide the higher consistency, but their two main disadvantages, namely storage requirements and computational cost, as well as the complexity costs of the required data structures should also be carefully considered.

## References

1. José A. Cotelo Lema, Ralf H. Güting. *Dual Grid: A New Approach for Robust Spatial Algebra Implementation*. Technical Report Informatik 268. FernUniversität Hagen. Germany. May 2000. To appear in the *Geoinformatica Journal*. Available at <http://www.fernuni-hagen.de/inf/pi4/bibliography.html>
2. E. Chan, R. Zhu. *QL/G - A Query Language for Geometric Databases*. TR CS-94-25, Univ. of Waterloo, 1994.
3. M. Egenhofer. *Spatial SQL: a Query and Presentation Language*. In IEE Transactions on Knowledge and Data Engineering. Vol 6, N°1, 1994
4. L. Forlizzi, R. H. Güting, E. Nardelli, M. Schneider. *A Data Model and Data Structures for Moving Objects Databases*. Proc. ACM SIGMOD Int. Conference on Management of Data, Dallas (Texas), pages 319-330. May 2000.
5. S. Grumbach, G. Kuper. *Tractable Recursion over Geometric Data*. In Intl. Conference on Constraint Programming, 1997.
6. S. Grumbach, P. Rigaux, L. Segoufin. *The Dedale System for Complex Spatial Queries*. In Proc. of the ACM SIGMOD Intl. Conference on Management of Data, pages 213-224. 1998.
7. S. Grumbach, P. Rigaux, M. Sholl, L. Segoufin. *Dedale: A spatial constraint database*. In Intl. Workshop in Database Programming Languages (DBPL'97), 1997.
8. R. H. Güting, M. H. Böhlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, M. Vazirgiannis. *A Foundation for Representing and Querying Moving Objects*. ACM Transactions on Database Systems (TODS), 25:1 (2000), pages 1-42.
9. R. H. Güting, T. de Ridder, M. Schneider. *Implementation of the ROSE Algebra: Efficient Algorithms for Realm-Based Spatial Data Types*. In Proc. of the 4th Intl. Symposium on Large Spatial Databases, pages 216-239. Portland, Maine. August 1995.
10. 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.
11. R. H. Güting, M. Schneider. *Realm-Based Spatial Data Types: The ROSE Algebra*. In VLDB Journal. Vol. 4(2), pages 100-143.
12. O. Günter. *Efficient Structures for Geometric Data Management*. In Lecture Notes in Computer Sciences LNCS 377, Springer-Verlag. 1988.
13. R. H. Güting. *Geo-Relational Algebra: A Model and Query Language for Geometric Database Systems*. In Proc. of Extending Database Technology, Venice, pages 506-527, 1988.
14. R. H. Güting. *Gral: An Extensible Relational Database System for Geometric Applications*. In Proc. of 15th VLDB, Amsterdam, pages 33-44, 1989.
15. R. H. Güting. *GraphDB: Modelling and Querying Graphs in Databases*. Proc. of the 20th Intl. Conference on Very Large Databases, Santiago, 1994, 297-308.
16. 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.
17. N. A. Lorentzos, J. R. Rios Viqueira, N. Tryfona. *Quantum-Based Spatial Extension to the Relational Model*. Dimitrios I. Fotiadis and Stavros D. Nikolopoulos (Eds.), *Advances in Informatics - Proc. of the 7th Hellenic Conference on informatics (HCI '99)*, World Scientific 2000, pp 188-199.
18. V. Muller, N. W. Paton, A. A. Fernandes, A. Dinn, M. H. Williams. *Virtual Realms: An Efficient Implementation Strategy for Finite Resolution Spatial Data Types*. In Proc. of the 7th Intl. Symposium on Spatial Data Handling - SDH'96, vol 2, pages 11B.1-11B.13. Delft, The Netherlands, 1996.
19. J. Paredaens. *Spatial Databases, the Final Frontier*. In G. Gottlob and M. Y. Vardi, editors, Proc. of the 5th Intl. Conference on Database Theory - ICDT'95. Lecture Notes in Computer Science LNCS 893, pages 14-32. Springer-Verlag, 1995.
20. Ph. Rigaux, M. Scholl. *Multiple Representation Modelling and Querying*. In IGIS94, 1994.
21. H. Samet. *The Design and Analysis of Spatial Data Structures*. In Addison-Wesley, 1990.