OPERATORS FOR CELL TUPLE-BASED SPATIOTEMPORAL DATA MODEL

Ale Raza

ESRI
380 New York Street, Redlands, California 92373-8100, USA
Tel.: +1-909-793-2853 (ext. 2009)
Fax: +1-909-307-3067
araza@esri.com

Commission IV, WG IV/1

KEY WORDS: Spatial, Temporal, Mathematics, Data Structures, Design, Modeling, GIS

ABSTRACT:

A spatiotemporal data model consists of attributes or classes, operations, and consistency constraints. Various models have been proposed by the geographic information system (GIS) community. These models could not be implemented because of various reasons. One of the reasons is that these models lack completeness and remain limited to the classes. This paper discusses the dynamic operators for the object-oriented cell tuple-based spatiotemporal data model (CTSTDM). The CTSTDM and its application in urban planning were published earlier by the author. Three main classes are defined in CTSTDM (i.e., spatial, attribute, and temporal). A spatiotemporal class is the aggregation of spatial and temporal classes. A spatiotemporal class is a super class of three classes (i.e., ZeroTCellClass, OneTCellClass, and TwoTCellClass). Operators pertaining to the object of these subclasses are elaborated in this paper. Emphasis is given to TwoTCellClass. Two types of operators can be defined as the objects of these classes: static and dynamic operators. Static operators do not affect the system's state (e.g., query operators). Dynamic operators change the state of the system (e.g., create, update, or delete operators). Unlike atemporal GIS, in a temporal GIS, objects may die or be killed, but they remain in the database with a valid time stamp indicating their life span. Therefore, four dynamic operators can be distinguished in spatiotemporal databases (i.e., Create, Kill, Reincarnate, and Delete). In spatiotemporal databases, the Kill operation is different from the Delete operation, as the latter is merely a purge operation. Updating a spatiotemporal object is a complex operation. Therefore, Kill is a protected operation, while the others are public or private. An object-oriented approach and a notion of point set topology are employed to design these operators in a systematic manner. Designing these operators in this fashion may pave the way to fill the gap between concepts, design, and implementation of a generic and functional temporal GIS.

1. INTRODUCTION

A data model consists of data members (classes or attributes), member functions (operations), and consistency constraints. Many spatiotemporal data models have been proposed in the past two decades. One main impediment for the implementation of these models has been the lack of completeness of these proposed data models. This problem is rooted in the inherent complexity of spatiotemporal data models. Most models are restricted to the classes or conceptual schema—one of the three main components of any data model. These classes are the first abstraction level for data modeling. The second component is the operation. Operations define the behavior of the model and act as an engine to run these models. The engines are guided by consistency rules—the third component of any data model.

These three components have been defined by the author in the object-oriented CTSTDM. The CTSTDM's classes, some operations, and consistency rules have been published (Raza and Kainz, 1999; Raza and Kainz, 2000a). The application of CTSTDM in urban planning was published later (Raza and Kainz, 2002; Raza and Kainz, 2000b). This model has been implemented by Jefferson County, Colorado, USA, for keeping track of the county's parcels system (Bochner, 2003).

Three main classes are defined in CTSTDM (spatial, attribute, and temporal). A spatiotemporal class is the aggregation of spatial and temporal classes. A spatiotemporal class is a super class of three classes (i.e., ZeroTCellClass, OneTCellClass, and TwoTCellClass). ZTC, OTC, and TTC are the objects of these three classes. The spatiotemporal class and TwoTCellClass are defined in §2. The spatiotemporal topology is preserved in a cell tuple structure. This cell tuple structure is discussed in §3.

Operations in spatial databases can be categorized as static (e.g., calculating area, length, orientation) and dynamic (e.g., adding new node, arc, polygon). Normally in atemporal GIS, three fundamental dynamic operations are applied (i.e., insert, delete, and update). Unlike atemporal GIS, in spatiotemporal databases (TGIS) objects may die or be killed, but they remain in the database with a certain time stamp indicating their life span. The fundamental dynamic operators are discussed in §4. How these operators are applied to TwoTCellClass is discussed in §5. The paper is concluded in §6.

2. SPATIOTEMPORAL CLASSES

The object of this class is defined in a space at time t. Formally, we can define these objects as follows:

An (open) *m*-tcell is a topological space homeomorphic to an open ball E^m of R^n (Euclidean *n*-space). A finite collection *k* of *m*-tcells is a TemporalCellComplex (TCC) if

- Different elements of k have disjoint interiors.
- For each *m*-tcell in *k,* the boundary of *m* is a union of elements of *k.*

- If a, $b \in k$, and $a \cap b \neq \varnothing$, then $a \cap b$ is a tcell and a union of elements of k, where tcell is either a or b (of different dimension) or a common face.

The discussion is restricted to $n = 2$ (2D space–time), where $m \in \{0,1,2\}$. Therefore, $R^{m,n}$ represents the object in E^m at time t, where m is the dimension of the object and n is the dimension of space where the object is located at time t, such that $n \geq m$.

As defined earlier, ZeroTCellClass, OneTCellClass, and TwoTCellClass are the subclasses of SpatioTemporalClass. In this paper we shall focus on TwoTCellClass. The object of this class is a TTC. A TTC is a two-dimensional object bounded by a closed cycle of ZTCs and OTCs. Any j-th TTC = set{OTC_1, OTC_2, …..OTC_i, ….OTC_n}, where i = {1,2,… n}. When i = 1, the first and last point object and the start and end ZTC are identical.

The data members of TwoTCellClass are a set of onetcells, 1-T (systemtime), area, perimeter, and parent (TwoTCell). The life of each TTC is depicted by 1-T [T_{From}, T_{Until}]. Like OTC, this object too can either be born or die. The birth and death times are represented by two point times, T_{From} and T_{Until}, respectively. A TTC must have one or more OTCs, and an OTC may have zero or two TTCs. Each TTC object may have one or more children, and each child (TTC) must have a parent (TTC).

When an n-tcell can be born or can die is an important decision. It is logical to investigate the situations where an n-tcell is changed as a result of the updating (insert or destroy) process in the spatiotemporal database. These operations are necessary for designing the algorithms of various operators.

While updating the data, an object (n-tcell) can intersect with another object. In the unified spatiotemporal data model, when a ZTC, OTC, or TTC is inserted, the following scenario can be expected:

- A ZTC may intersect with ZTC, OTC, or TTC.
- An OTC may intersect with ZTC, OTC, or TTC.
- A TTC may intersect with ZTC, OTC, or TTC.

There are nine possibilities when an n-tcell at time T1 may intersect with an n-tcell at time T2. In each case, there are various possibilities (e.g., ZTC may intersect at the boundary of OTC or the interior of OTC). Cases in which a TTC intersects with a TTC are discussed in the paper. The three topological invariants of spatial objects (n-tcells) are boundary, interior, and exterior.

This point-set topology approach is employed to analyze these intersections. Only the boundary (∂) and interior ($^\circ$) of OTC and TTC are considered in order to investigate these intersections. The intersection at the exterior of any n-tcell is straightforward. The boundary of ZTC is empty ($\varnothing$). A similar approach (i.e., point-set topology) has been employed by Egenhofer *et al.* (1994) to identify and/or compare topological relationships between n-dimensional objects embedded in R^n.

3. TEMPORALCELLTUPLECLASS

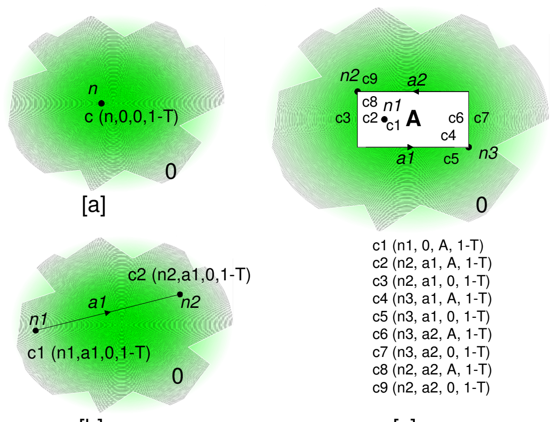
Because the object is defined as a spatiotemporal object, the topological relations could be defined as spatiotemporal topological relations (i.e., the spatial relations that are valid over time). In the temporal cell complex, *Intra* cell complex relations (i.e., relations between the cells in the cell complex) can be described using boundary and co-boundary relations. The boundary (∂) of an n-tcell is its (n-1) faces at time t. The co-boundary (Φ) of an n-tcell produces the (n+1) cells incident with n-tcell at time t. The boundary and co-boundary relations capture two types of topological relationships (i.e., adjacency and containment). Relations between spatial objects can be found based on boundary/co-boundary relations between cells. The boundary and co-boundary relations are encapsulated in a simple temporal cell tuple structure, which is an extension of the cell tuple structure of Brisson (1990). A cell tuple T is an (n+1)-tuple of cells {c_0, c_1, c_2,....,c_n}, where any i-cell is incident with a (i+1)-cell.

The object of TemporalCellTupleClass has a unique tuple-ID and a unique combination of ZTC, OTC, and TTC. Each tuple must have a ZTC, zero or one OTC, and zero or one TTC. Therefore, a temporal cell tuple structure encapsulates the spatiotemporal topology of each spatiotemporal object. A temporal cell tuple (TCT) is a set of C and T.

TCT = {C, T}
where C is a set of cells
$C = \{c_0, c_1, c_2, \ldots, c_n \mid c_i \in TCC\}$ and
T is a time interval (1-T)
$T = \{T_{From}, T_{Until} \mid (T_{From} < T_{Until}) \wedge (T_{From}, T_{Until} \in ST)\}$
and
TCC = TemporalCellComplex

Therefore,
$TCT = \{c_0, c_1, c_2, \ldots, c_n, T_{From}, T_{Until}\}$

The process of assigning the cell tuples to a ZTC is illustrated in Figure 1.



c1 (n1, 0, A, 1-T)
c2 (n2, a1, A, 1-T)
c3 (n2, a1, 0, 1-T)
c4 (n3, a1, A, 1-T)
c5 (n3, a1, 0, 1-T)
c6 (n3, a2, A, 1-T)
c7 (n3, a2, 0, 1-T)
c8 (n2, a2, A, 1-T)
c9 (n2, a2, 0, 1-T)

Figure 1. Process of assigning temporal cell tuples to spatiotemporal cells of dimensions ($0 \leq n \leq 2$).

4. OPERATORS

Two types of operators can be defined (i.e., static and dynamic). Static operators do not affect the system's state or the status of spatiotemporal objects (e.g., query operators calculate the length, area, time period, boundary, or co-boundary). These operators are associated with TemporalCellTupleClass. On the other hand, dynamic operators change the state of the system or the status of the spatiotemporal objects (e.g., creating, deleting, or updating an n-tcell). Normally in atemporal GIS, three fundamental dynamic operations are performed (i.e., create, delete, and update). These operators are associated with PointClass, ZeroTCellClass, OneTCellClass, and

TwoTCellClass. Unlike in atemporal GIS, in a TGIS objects may die or be killed, but they remain in the database with a certain time stamp indicating their life span. As mentioned earlier, any *n*-tcell object can be born or can die. Therefore, four fundamental dynamic operators can be distinguished in spatiotemporal databases (i.e., **Create, Kill, Reincarnate,** or **Delete [Destroy]**). These operators are associated with objects (ZTC, OTC, or TTC). In spatiotemporal databases, the Kill operation is different from the Delete operation, as the latter is merely a purge operation. Updating spatiotemporal objects is complex; any update operation affects the other objects, particularly in the unified approach. Any spatial change is the result of the creation (birth) and/or destruction (death) of an *n*-tcell. Kill is a protected operation, while the others are public or private.

- The **Create** operator is equivalent to the usual insert operators. The task of this operator is to create a new object and/or update an existing object. This operator specifies the time stamp [start, *] of each spatial object, where the upper bound of the time interval is undefined (*). All objects with [start, *] time stamps are called active objects.
- The **Kill** operator kills the spatiotemporal objects by defining the upper bound of the time interval. After being killed, objects are called inactive objects. These objects remain in the database only for the query purpose or Reincarnate operator. Therefore, the upper bound (*) is replaced by current system time.
- The **Delete** operator permanently deletes the spatiotemporal objects from the database. Therefore, they are no longer available for any type of operation (static or dynamic).
- The **Reincarnate** operator turns an inactive object into an active object by replacing the upper bound of the time interval to (*).

5. OPERATORS FOR TWOTCELLCLASS

The aforementioned operators (Create, Kill, Delete, and Reincarnate) are discussed and applied to the TTC in the following section.

5.1 Operation Create (Insertion of TTC)

The Create operation for TTC is a recursive operation, starting from the insertion (creation) of the boundary of TTC (i.e., OTC and boundary of OTC, which is ZTC). This operation can be viewed from three perspectives (i.e., a TTC can intersect with ZTC, OTC, and TTC). The cases in which TTC intersects with TTC are discussed here; the other two cases (TTC–OTC and TTC–ZTC) are semisymmetric (because the geometry is the same, while the spatiotemporal topology is different) to the OTC–TTC and ZTC–TTC intersections. These cases are discussed earlier (Raza and Kainz, 2000a).

Let TTC (A) and TTC' (B) be a TTC at time T1 and T2, respectively. Using the point-set approach, a 2 x 2 intersection matrix can be constructed:

$$TT_{14} = \begin{pmatrix} \partial A \cap \partial B' & \partial A \cap {}^{\circ}B \\ {}^{\circ}A \cap \partial B & {}^{\circ}A \cap {}^{\circ}B \end{pmatrix}$$

Out of 16 possible intersections, only seven (TT 1, 7, 9, 10, 11, 13, and 14) are valid, while the last one (16) is a nonintersection case and TT 11 and 12 are symmetric.

1]	∂A intersects with ∂B
7]	${}^{\circ}A$ intersects with $\partial {}^{\circ}B$
9]	${}^{\circ}B$ intersects with $\partial {}^{\circ}A$
10]	$\partial {}^{\circ}A$ intersects with $\partial {}^{\circ}B$
11]	∂A intersects with $\partial {}^{\circ}B$ and ${}^{\circ}A$ intersects with ∂B
13]	∂B intersects with $\partial {}^{\circ}A$ and ${}^{\circ}A$ intersects with ${}^{\circ}B$
14]	∂A intersects with $\partial {}^{\circ}B$ and ${}^{\circ}A$ intersects with ${}^{\circ}B$

Egenhofer (1993) derived binary topological relations between two regions using this approach. Figure 5 is a general illustration of TTC–TTC intersections; each intersection may have various combinations, some of which are associated with valid intersections and are discussed in the following sections.

5.1.1 Boundary of TTC Intersects With Boundary of TTC' (TT-1):

Five combinations can be realized in this intersection.

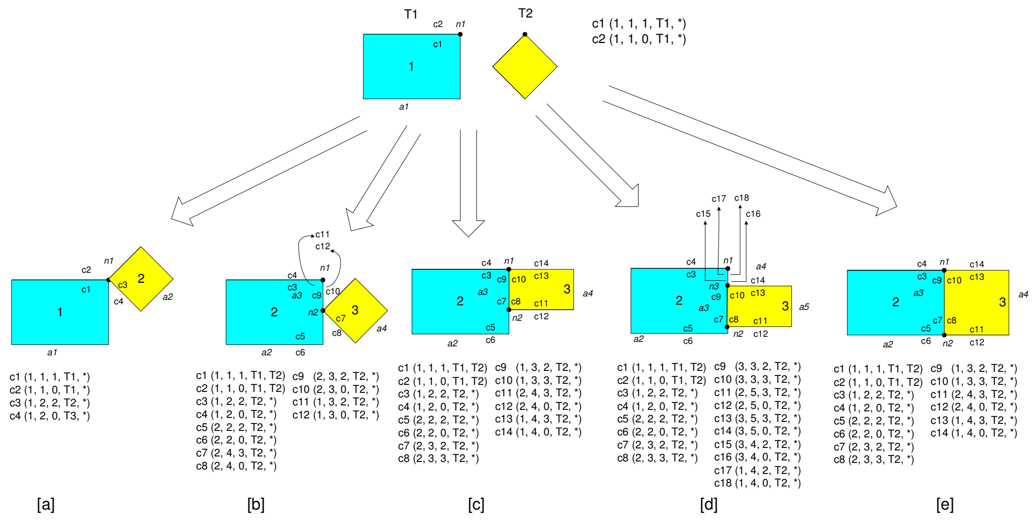


Figure 2. Create TTC: Boundary of TTC intersects with boundary of TTC.

[a] Both TTCs intersect at ZTC (Figure 2[a]):
Create OTC (*a2*), TTC (2), and TCTs (c3 and c4).

[b] OTC of TTC (1) intersects with ZTC of TTC (2) (Figure 2[b]):
Kill OTC (*a1*), TTC (1), and TCTs (c1 and c2). Create ZTC (*n2*), OTCs (*a2, a3,* and *a4*), TTCs (2 and 3), and TCTs (c3, c4,….,c12).

[c] ZTC and OTC of TTC (1 and 2) intersect (Figure 2[c]):
Kill OTC (*a1)*, TTC (1), and TCTs (c1 and c2). Create ZTC (*n2*), OTCs (*a2, a3,* and *a4*), TTCs (2 and 3), and TCTs (c2, c3,….,c14).

[d] OTC of TTCs (1 and 2) intersects (Figure 2[d]):
Kill OTC (*a1)*, TTC (1), and TCTs (c1 and c2). Create ZTCs (*n2* and *n3*), OTCs (*a2, a3, a4,* and *a5*), TTCs (2 and 3), and TCTs (c2, c3,….,c18).

[e] ZTC and OTC of TTCs (1 and 2) intersect (Figure 2[e]):
Kill OTC (*a1)*, TTC (1), and TCTs (c1 and c2). Create ZTC (*n2*), OTCs (*a2, a3,* and *a4*), TTCs (2 and 3), and TCTs (c2, c3,….,c14).

The configuration formed in Figure 2[a], [b], [c], [d], and [e] is called a 2-temporal complex (2-TC) or 2D spatiotemporal object (TSTO). A 2-TC is a collection such that each TTC in 2-TC is connected through a common face.

5.1.2 Interior of TTC Intersects With Boundary–Interior of TTC' (TT-7): When the interior of TTC (1) intersects with the boundary of TTC (2), then the following actions are taken (Figure 3):

Kill TTC (1) and TCT (c2). Create ZTC (*n2*), OTC (*a2*), and TCTs (c3, c4, and c5).

TCT c2 is replaced by c3, because the co-boundary of OTC (*a1*) is changed while the OTC (*a1*) remains unchanged. At time T1, the co-boundary of *a1* was <0,1> and at time T2 it was changed to <0,2>. This is one of the advantages of the implicit topology storage approach. If the topology is stored in an implicit fashion, then the object also has to be updated in order to update the topology, because topology is associated with the spatial object (e.g., in ArcGIS® the co-boundary [left and right polygon] information is associated with arc).

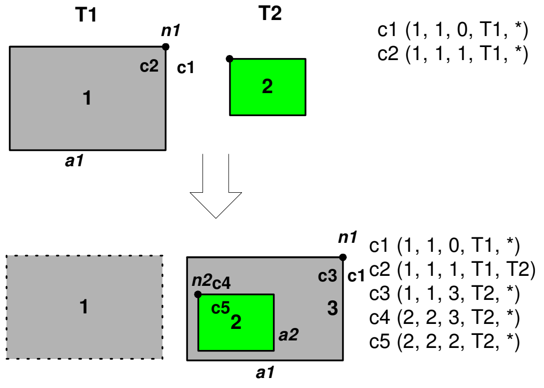


Figure 3. Create TTC: Interior of TTC intersects with boundary–interior of TTC'.

5.1.3 Interior of TTC' Intersects With Boundary–Interior of TTC (TT-9): When the boundary–interior of TTC (1) intersects with the interior of TTC (2), then the following actions are taken (Figure 4):

Kill TCT (c2). Create ZTC (*n2*), OTC (*a2*), and TCTs (c3, c4, and c5).

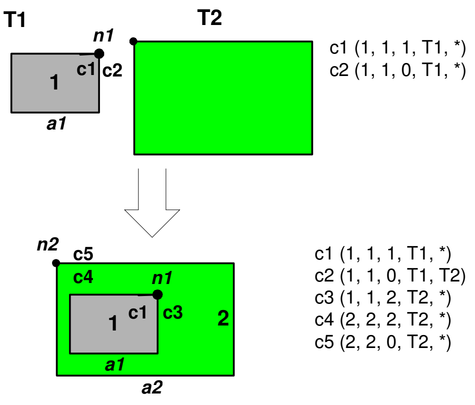


Figure 4. Create TTC: Interior of TTC' intersects with boundary–interior of TTC.

In this case, TTC (1) remains unchanged but its spatiotemporal topology is adjusted (i.e., TCT c2 is replaced by c3).

5.1.4 Boundary–Interior of TTC Intersects With Boundary–Interior of TTC' (TT-10): The operation rejects the TTC (2) because this cell is identical or the same as TTC (1).

5.1.5 Boundary of TTC Intersects With Boundary–Interior of TTC', and Interior of TTC Intersects With Boundary of TTC' (TT-11):

Kill TTC (1), OTC (*a1*), and TCT (c1 and c2). Create ZTCs (*n2, n3,* and *n4*), OTCs (*a2, a3, a4, a5, a6,* and *a7*), TTCs (2, 3, and 4), and TCTs (c3, c4,....,c26).

The process is shown in Figure 5.

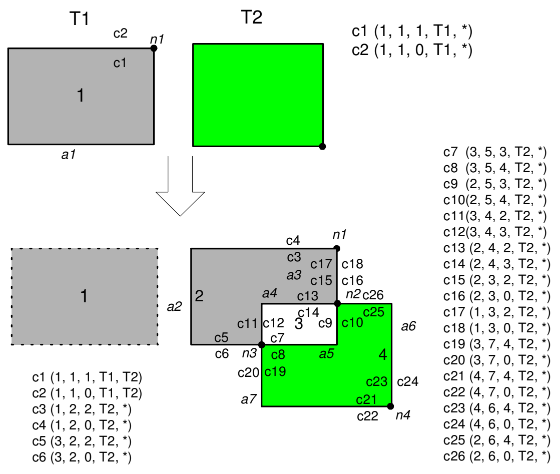


Figure 5. Create TTC: Boundary of TTC intersects with boundary–interior of TTC', and interior of TTC intersects with boundary of TTC'.

5.1.6 Interior of TTC Intersects With Boundary–Interior of TTC', and Boundary of TTC Intersects With Boundary of TTC' (TT-13): The boundaries of TTC 1 and 2 are their OTC. This can be considered as OTC–OTC intersections, where OTCs can intersect in many ways. Three cases are illustrated here.

[a] Boundary of OTC intersects with interior of OTC (Figure 6[a]).

Kill TTC (1) and TCT (c2). Create OTC (*a2*), TTCs (2 and 3), and TCTs (c3, c4, and c5).

TCT c2 is replaced by c3 because the co-boundary of OTC (*a1*) is changed to 2 (at time T2) from 1 (at time T1).

[b] Interior of OTC intersects with interior of OTC (Figure 6[b]).

Kill TTC (1), OTC (*a1*), and TCTs (c1 and c2). Create ZTC (*n2*), OTCs (*a2, a3,* and *a4*), TTCs (2 and 3), and TCTs (c3, c4,....,c12).

[c] Both boundary and interior of OTC intersect each other (Figure 6[c]).

Kill TTC (1), OTC (*a1*), and TCTs (c1 and c2). Create ZTC (*n2* and *n3*), OTCs (*a2, a3, a4,* and *a5*), TTCs (2 and 3), and TCTs (c3, c4,....,c18).

These examples show that a different number of TCTs are generated depending on the geometric configurations of the temporal cells, although topologically they are all the same.

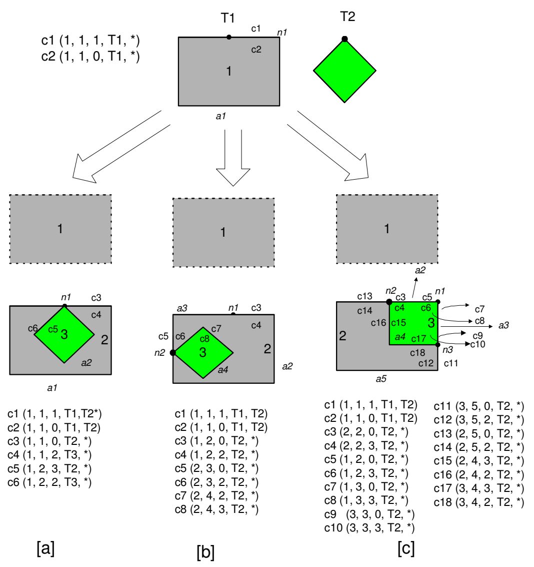


Figure 6. Create TTC: Interior of TTC intersects with boundary–interior of TTC', and boundary of TTC intersects with boundary of TTC'.

5.1.7 Boundary of TTC Intersects With Boundary–Interior of TTC, and Interior of TTC Intersects Interior of TTC' (TT-14): This is similar to the previous case (TT-13), except the TTC at time T1 is not killed (Figure 7).

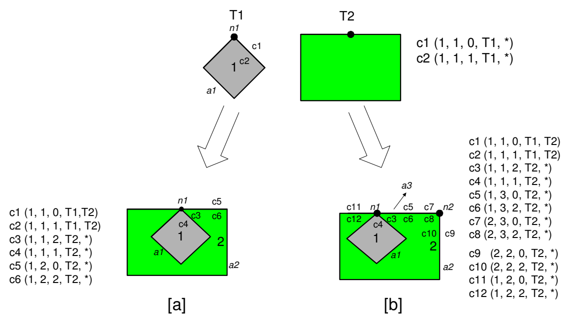


Figure 7. Create TTC: Boundary of TTC intersects with boundary–interior of TTC, and interior of TTC intersects interior of TTC'.

5.2 TTC Kill Operator (↓)

While applying Kill operators to TTC, two scenarios can be realized.

[a] The face of TTC is not shared by other TTCs or isolated TTCs (Figure 8[a]):
Kill ZTC (*n1*), OTC (*a1*), TTC (1), and TCTs (c1 and c2).

All the faces and TTC itself are killed.

[b] The face of TTC is shared by another TTC (Figure 8[b]): Kill OTC (*a1*), TCT (1), and TCTs (c5, c6, c7, c8, c10, and c12).

All the faces are killed except common face(s).

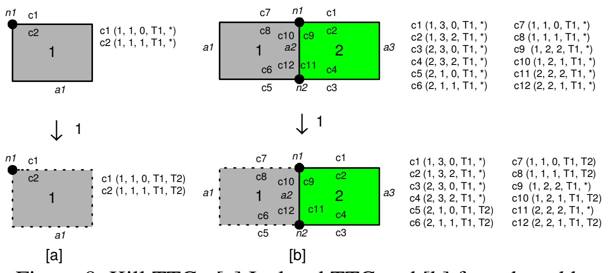


Figure 8. Kill TTC: [a] Isolated TTC and [b] face shared by another TTC.

5.3 Delete (⇓) and Reincarnate (↑) Operators

The Delete operator is based on the Kill operator (i.e., the same algorithm is applied to the Delete operator as applied to the Kill operator). Once the objects (*n*-tcells) are killed, they can be purged from the database. The Delete operator purges the database by permanently deleting *n*-tcells instead of making them inactive. Therefore, these cells are no longer available for the Reincarnate operator. The Reincarnate operator turns an inactive cell into an active cell by replacing the upper bound (ST_Until) of the time interval with a null value. One example is considered here to demonstrate the function of the Reincarnate operator. This operator is pragmatic in retroactive changes. For example, at time T1, there was one TTC (A); at time T2, two new TTCs (B and C) were created. The TTC (A) has been killed because of the Create operation at time T2. Scenario 1 is shown in Figure 9. At time T3, it was realized that the TTCs (B and C) had been wrongly created (wrong configuration). Actually they have to be created in the fashion shown in scenario 2, which is the actual configuration (Figure

10). In scenario 1, at time T3, the configuration of B and C (scenario 2) could not be achieved because the TTC (A) is no longer an active object. Prior to achieving the scenario 2 configuration as shown at time T2 in scenario 1, TTCs B and C have to be killed and TTC A must be reincarnated. These steps are explained as follows and demonstrated in Figure 11.

At time T2:
[a] Wrong configuration, which needs to be corrected.

At time T3:
[b] Delete (⇓) TTCs (B and C) and corresponding faces and TCT (not shown in the figure for simplification reasons).
[c] Reincarnate (↑) TTC (A), which includes the reincarnations of TCTs (not shown for simplification reasons).
[d] Create TTC (B) to achieve desired temporal cell complex configuration.

The example shown here for TTC can be applied for ZTC and OTC. However, more work is needed to analyze the scenario when applying/designing the Reincarnate operators for ZTC and OTC.

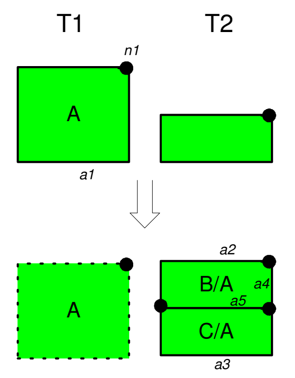


Figure 9. Scenario 1: Wrong configuration of TTCs (B and C) at time T2.

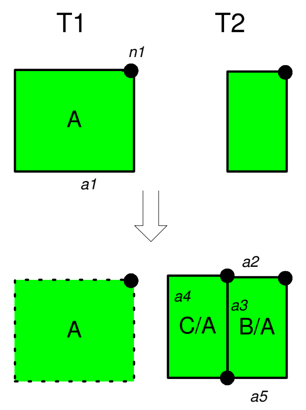


Figure 10. Scenario 2: Actual configuration of TTCs (B and C) at time T2.

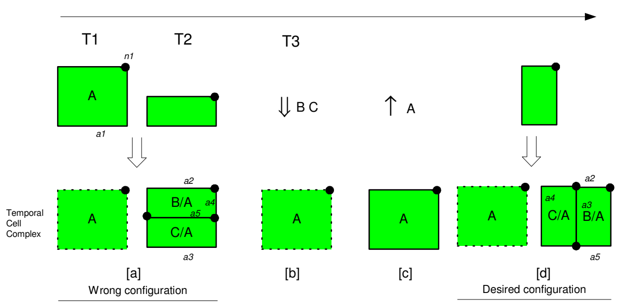


Figure 11. Reincarnate (↑) operator: Demonstration of retroactive change

6. CONCLUSION

Designing a spatiotemporal data model is a complex process. In this paper, four fundamental dynamic operators (Create, Kill, Delete, and Reincarnate) are identified and applied to the TwoTCellClass of CTSTDM. It has been demonstrated that by employing object-oriented concepts and mathematical theory of point-set approach, the complexity can be diluted. The Create operator has been applied to the cases in which TTC intersects with TTC. Based on empty and nonempty combinations, seven valid cases are discussed. The Kill operator has been applied for shared-face and isolated TTC. It has been demonstrated how the TTC can be killed and reincarnated. In all operations the spatiotemporal topology is preserved in the cell tuple structure. Designing operators in this fashion may pave the way to fill the gap between concepts, design, and implementation of a functional temporal GIS.

7. REFERENCES

Bochner, P., 2003. ADT (Application Development Trends) 2003 Innovator Awards Component-based Development Winner, "Objects Keep Track of County Addresses", USA. http://www.adtmag.com/article.asp?id=7448 (accessed 13 April 2004).

Brisson, E., 1990. *Representation of d-dimensional geometric object*, Ph.D. thesis, University of Washington.

Egenhofer, J.M., E. Clementini, and P. Felice, 1994. Topological relations between regions with holes, *IJGIS (International Journal of Geographical Information Science),* Vol. 8, No. 2, pp. 129-142.

Egenhofer, J.M., 1993. A model for detailed binary topological relationships, *Geomatica*, Vol. 47, No. 3 and 4, pp. 261-273.

Raza, A., and W. Kainz, 2002. An object-oriented approach for modeling urban land use changes. *Journal of the Urban and Regional Information Systems Association (URISA)*, Vol. 14, No. 1, pp 37-55.

Raza, A., and W. Kainz, 2000a. Designing operators for object-oriented spatio-temporal data model. *The International Society for Photogrammetry and Remote Sensing (ISPRS),* Amsterdam, the Netherlands, Vol. XXXIII, Part B4, pp. 863-870.

Raza, A., and W. Kainz, 2000b. An object-oriented approach for modeling urban land use changes. In: *The Proceedings of the Urban and Regional Information Systems Association,* Florida, USA, pp. 20-25.

Raza, A., and W. Kainz, 1999. Cell tuple based spatio-temporal data model: an object oriented approach. In: *The Proceedings of the Eighth ACM Conference on Information and Knowledge Management (CIKM'99) and Symposium on Geographic Information Systems (GIS'99),* Kansas City, Missouri, USA, pp. 20-25.