Karimipour, Farid, M. R. Delavar, A. U. Frank, and H. Rezayan. "Point in Polygon Analysis for Moving Objects." Paper presented at the The 4th Workshop on Dynamic & Multi-Dimensional GIS, Glenmorgan, Wales, 5.-8.09.2005 2005.

F. Karimipour [a], M. R. Delavar [a]*, A. U. Frank [b], H. Rezayan [a]

[a] Dept. of Surveying and Geomatic Eng., Eng. Faculty, University of Tehran, Tehran, Iran
(fkarimipr, mdelavar, rezayan)@ut.ac.ir

[b] Dept. of Geo-Information E-127, Technische University Wien, Gusshausstr. 27-29, A-1040, Vienna, Austria
frank@geoinfo.tuwien.ac.at

KEY WORDS: Algorithms, Dynamic, GIS, Mathematics, Modelling

ABSTRACT:

The decision whether a moving object is inside a polygon or not is a function of time. This is an important and instructive example problem to discuss a general method to deal with temporal data in GIS. Recently, some efforts have been done to handle temporal dimension of our space effectively both in our theoretical and commercial approaches. However, existing commercial GISs have only very limited support for it. In this situation, GI theory is investigating an appropriate solution through formalizing the utilization of time on the basis of mathematical and computer sciences. This formalization is carried out by definition of spatial and temporal concepts, operators, and processes in GI as abstract algebras, which are mapped together using morphisms. The achieved results have to be more advanced by testing different hypothesis. This idea has been implemented for time lifting of issues related to moving objects in this paper and the mentioned approaches are used for hypothesis of integrating static and dynamic point in polygon analysis into a unique algorithm. The conclusions out coming from this work certify validity of these approaches for point in polygon analysis for moving objects. The results will be generalized to the rest, as further steps of this research.

1. INTRODUCTION

Time is inherently linked to geospatial concepts (Egenhofer and Mark 1995). However, existing commercial geospatial information systems (GISs) still have shortcomings handling time. In this situation, GI theory must show how to deal with temporal aspects—including changing values, processes, etc. (Frank, 2005). Then GI theory is directed toward formalizing time utilization on the basis of mathematical and computer sciences that are implemented into GISs. This formalization is carried out by definition of spatial and temporal concepts, operators, and processes in GI as abstract algebras that are mapped together using morphisms. Morphisms from non-temporal to temporal domains are usually mentioned as time lifting in recent researches.

This idea could be applied for time lifting of moving objects related issues. Moving objects are entities in a solid geometry space that are substituted with points, so they are mentioned as moving points, too. Hence time lifting has to be applied for analyses carried out on domain of static point sets to moving point sets. Point in polygon analysis is one of the basic analyses on point sets that are proposed to be time lifted in this paper.

The paper is composed of seven sections. Section 2 describes some basic issues about time utilization in GISs. In Section 3, the methodology of the paper is introduced that deals with how to time lift operations and analyses and also how they could be implemented. Moving points and algebraic definitions of their point in polygon analysis are illustrated in Sections 4 and 5, respectively. Section 6 represents the results of paper as a case study of a point in polygon analysis for a test set of moving points. Finally, Section 7 provides conclusions and future work.

2. TIME IN GIS

The decision whether a moving object is inside a polygon or not is a function of time. This is an important and instructive example problem to discuss a general method to deal with temporal data in GIS. Recently, some efforts have been done in the GI community to handle temporal dimension of our space effectively both in our theoretical and commercial approaches. Despite many efforts and researches carried out dealing with the temporal domain of GI science, some deficiencies are observable as follows:

- Considering time as a discrete or partial continuous property of our world while our unique physical reality is governed by differentiable laws (Frank, 2003).
- Underestimation of different behaviours of models that are used in GIS (e.g., network, object, and field) that resulted in development of some temporal context-based viewpoints that can not be generalized (Frank, 2005).

All these issues could somehow be interpreted as dominance of analytical treatments that are affected by the limitation of our computer systems (e.g., its discrete and limited numerical system) and lack of a comprehensive temporal ontology. The proposed solution is dealing with an algebraic treatment of different models in GIS independently and provides a basis for their integration, towards development of a generalized and implementable temporal ontology.

The mentioned solution is considered in some researches as the basic form ontology illustrated for space (SNAP) and time (SPAN) by Grenon and Smith (2004) and multi-tier ontology

* Corresponding author

presented by Frank (2003). Besides that, introduction of mathematical and algebraic approaches on the basis of category theory and utilization of functional programming environments (Herring et al., 1990) (which allow variables stand for functions) has provided some promising future for temporal GIS.

3. METHODOLOGY

One of the fundamental concepts on the basis of the proposed advancements for incorporation of time in GIS analyses is lifting them from the domain of values to the domain of changing values (Frank, 2005). It is possible using mediums for these domains that could support the lifting. Categories provide the required medium. They are defined in category theory that is a branch of mathematics.
A category is a collection of primitive element types, a set of operations upon those types, and an operator algebra that is capable of expressing the interaction between operators and elements (Herring et al., 1990).

A categorical viewpoint demonstrates that semantics of operations are independent of the representations they are applied to (Frank, 2005). Category theory gives us a very high level abstract viewpoint: instead of discussing the properties of individual objects, we directly address the properties of the operations. This corresponds to the interest in geography, where the discussion concentrates on processes that occur in space, not on the collection of locations and properties of spatial objects (Frank, 2005). These properties enable us to have a function between two categories with the same internal structure.
A functor, or more properly morphism, is the process that associates elements and operations from one category to another that preserves the operator algebra (Herring et al., 1990).

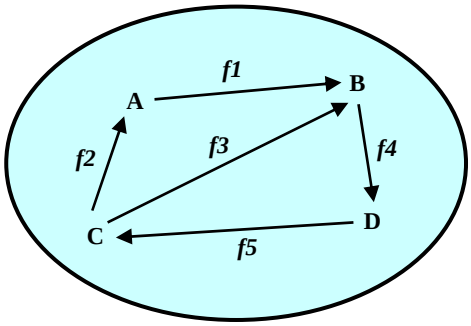


Figure 1. A category with its elements (Karimipour, et al., 2005)

Static and dynamic GIS domains have the same internal structure and the only difference between them is their dimensions. Since category theory concentrates on the processes instead of properties of objects, it seems an appropriate candidate to associate static and dynamic GIS domains and their objects and processes. From this viewpoint, the time lifting is a functor from a static category to a dynamic one.
The categories used here for time lifting are field categories that consist of fields as their objects and field homomorphism as their morphism. Given two groups *(G, +, *)* and *(H, ++, **)*, a group homomorphism from *G* to *H* is a function h: $G \rightarrow H$ such that for all *u* and *v* in *G* it holds that:

```
> h(u + v) = h(u) ++ h(v)
> h(u * v) = h(u) ** h(v).
```
(1)

The identities and inverses are mapped through this function.
One of the objects of these categories used here is field over rational numbers. A rational number is a ratio of two integers usually written as *a/b* where *b* is not zero. The set of all rational numbers is denoted by ***Q*** that is a dense subset of real numbers and totally ordered. Being a dense subset means that between any two rationals, there sits another one (in fact infinitely many other ones). So float numerical system of computers is substituted with the rational numerical system. Using the rational numbers we could get rid of round off error and dealing with our operations as continuous ones.
On the other hand, regarding the above-mentioned definition of functors, they are functions that have other functions as argument and achievement of such concepts is not part of the core structure of common programming languages such as C++. Implementation of both functors and rational numerical system is possible using a functional programming language. In a functional programming language every thing is a function that can accept a function as an input and produce a function as an output, too. Haskell is one of the functional programming languages that support our requirements (Haskell website).
The infinitely defined the Integer type in Haskell provides us an infinite rational number system. In this system time is defined as an infinite rational number (2).

```
> Type Time =Ratio Integer.
```
(2)

The changing version of any value is defined as a function from time to the value (3).

```
> type Changing v = Time -> v.
```
(3)

Then a rational changing value is defined as (4)

```
> Changing (Ratio Integer).
```
(4)

The functors for time lifting are defined as *lift0*, *lift1, lift2* and *lift3* to lift operators with zero, one, two, and three parameters functions, respectively (5).

```
> lift0 a       = \ t → a t
> lift1 op a    = \ t → op (a t)
> lift2 op a b  = \ t → op (a t) (b t)
> lift3 op a b c = \ t → op (a t) (b t) (c t).
```
(5)

These functors add time to input functions as parameter *t*. Lifting for operators with more arguments can be done in a similar way.
Some of the objects in a category are simple operators that can be lifted using the above-mentioned lifting process. Having these lifted operators, the functions that used them will be lifted automatically. In other words, this process is lifting simple operators of a category instead of functions that used them.

4. MOVING OBJECTS

Moving objects are entities in a solid geometry space and it is important for applications that keep track of cars, aircraft, or similar objects (Frank and Gruenbacher, 2001). These objects are substituted with points and are also mentioned as moving points.

Various operators are used for moving object operators according to the kinds of moving object and the application areas adopted (Ryu and Ahn, 2001). Seeking the position of a moving object at specific time, obtaining the distance between moving points at specific time, specifying the region where a moving point lies in it at specific time and obtaining the time when a moving object has a minimum or maximum value are some of the most major instances of them.

Movement of moving objects can be divided to coordinate elements. With no loss of generality, we focus on 2D moving objects. In this case, a moving object has two coordinate elements that are functions of time.

Points, edges, and polygons are defined in Haskell code as algebraic data types (6):

```
> data Point a = Point a a
> data Edge a = Edge (Point a) (Point a)
> data Polygon a = Polygon [Edge a].
```
(6)

The first statement represents a parametric 2D point. The second statement defines a straight line segment with its start at end points. The last statement depicts a polygon as a list of edges that construct the polygon boundary.

Then, basic operations for *point* data type are defined into class *points* (7).

```
> class Points p s where
>   x, y :: p s→ s
>   xy :: s → s → p s
>   (+) :: p s → p s → p s
>   (-) :: p s → p s → p s.
```
(7)

This class contains operations for extracting *x* and *y* coordinates, a point constructor, plus and minus operators. The class *Points* is overloaded for point data type in both static (8) and moving (9) modes. For moving points overloading just the plus and minus operators are required to be overwritten:

```
> instance Point a where
>   x    (Point x1 y1) = x1
>   y    (Point x1 y1) = y1
>   xy   x1 y1         = Point x1 y1
>   (+)  (Point x1 y1) (Point x2 y2) = Point (x1 + x2) (y1 + y2)
>   (-)  (Point x1 y1) (Point x2 y2) = Point (x1 - x2)  (y1 - y2).
```
(8)

```
> instance Changing Point a where
>   (+) = lift2(+)
>   (-) = lift2(-).
```
(9)

These operators are counterpart of static plus and minus operators. In addition, the similar operator symbols (that is + and -) are used for static and dynamic numbers and points through use of polymorphism mechanism means that a simple function can be applied to a variety of argument types.

x, *y* and *xy* operators are independent on type of point and can be used for both static and dynamic points. The other basic operations such as equality and ordering of points is defined and lifted similarly.

This simple lifted operation can be integrated for implementation of further complicated analyses and this sequence can be continued.

One of the simplest analyses carried out in point sets for delineation of their basic geometrical boundary is point in polygon analysis that is lifted in next section.

5. IMPLEMENTATION OF POINT IN POLYGON ANALYSIS FOR MOVING POINTS

Point in polygon analysis is one of the most frequently used queries in GIS. Given a map and a query point *q* specified by its coordinates, point in polygon analysis finds the region of the map containing *q*. A map, of course, is nothing more than a subdivision of the plane into regions.

Different algorithms are provided for point in polygon analysis with different complexities and performances (Berg et al., 2000).

Point in polygon analysis is implemented here following one of the simplest algorithms based on counting the number of intersections between an arbitrary ray passing from the target point and a region boundary (Figure 1). If the number of intersections is odd, the point is inside the region (Burrough and McDonnell, 1998). In case that the point lays exactly on the same y value of a node of polygon is excluded here.

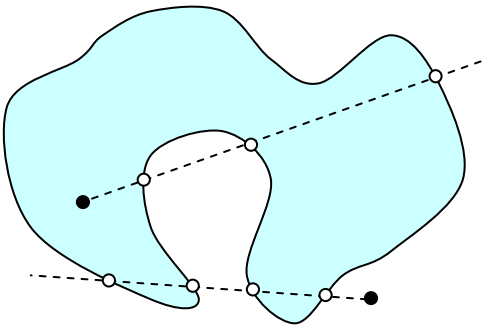


Figure 1. Point in polygon algorithm

At this point, implementation of static and moving point in polygon analysis is carried out by constructing the static algorithm discussed. The implemented algorithm is capable of operating on both static and moving points as they are both overloaded for basic operators.

In implemented algorithm, the intersecting ray from a point is limited to a horizontal ray. Finding intersections of a region with a horizontal line is reduced to filtering edges of the region boundary.

The intersection for a horizontal line ($y=a$) and one straight edge is defined here as checking the incidence of start (s) and end (e) points in different sides of the horizontal line (10).

```
> (ys<a<ye) or (ye<a<ys).
```
(10)

The Haskell code for this conditional inequality is represented in *isEdgeIntersected* function (11).

```
> isEdgeIntersected :: Point a → Edge a → Bool
> isEdgeIntersected point edge =                                    (11)
    isNumBetween (y (startNode edge)) (y (endNode edge)) (y point).
```

Also, the above-mentioned filtering is implemented into the *isPointInPolygon* function that proceeds by counting process and checking for oddness (12).

```
> isPointInPolygon :: Point a → Polygon a→ Bool
> isPointInPolygon point polygon =                                  (12)
    odd (length (filter (isEdgeIntersected point)  polygon.
```

The *pointInPolygons* function maps the previous function over a list of polygons (13).

```
> pointInPolygon :: Point a -> [Polygon a] -> [Polygon a]           (13)
> pointInPolygon pt pls = filter (isPointInPolygon pt) pls.
```

Finally, the *pointsInPolygons* function generalizes our functions for mapping *pointInPolygons* over a list of moving points (14).

```
> pointsInPolygons :: [Point a] -> [Polygon a] -> [[Polygon a]]
> pointsInPolygons pts pls = map (pointInPolygon' pls) pts where
    pointInPolygon' pls pts = pointInPolygon pts pls.               (14)
```

6. CASE STUDY

The implemented point in polygon analysis is used for a case study consisting of a set of static regions denoted as *polygons* (Figure 2) and a collection of five moving points over these regions (15).

```
> tPt1, tPt2, tPt3, tPt4, tPt5 :: Point (Changing (Ratio Integer))
> tPt1 = Point  (\t ->   50 * t - 500)   (\t ->   15 * t + 200)
> tPt2 = Point  (\t -> (-50)* t + 600)  (\t ->   30 * t - 400)
> tPt3 = Point  (\t -> (-10)* t - 200)   (\t ->   65 * t - 700)     (15)
> tPt4 = Point  (\t ->   35 * t - 20 )    (\t -> (-40)* t + 800)
> tPt5 = Point  (\t ->   10 * t  - 300)   (\t ->   40 * t - 200)
> tPoints = [tPt1, tPt2, tPt3, tPt4, tPt5].
```

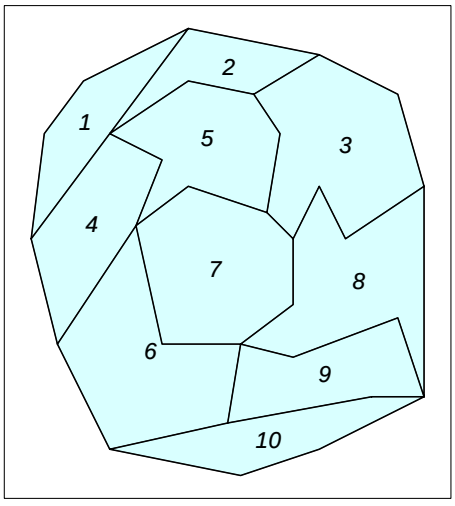


Figure 2. The study area

The dynamic point in polygon analysis is defined here as (16):

```
> PinP = pointsInPolygons tPoints Polygons.                         (16)
```

PinP is a function with one time parameter. In other words, *PinP t* returns list of the regions where the moving points are in them for instance *t*. The results of using the *PinP* for these moving points for three different instances are shown in Table 1 and Figure 3.

Point ID	t=0	t=10	t=20
1	(-500,200)	(0,350)	(500,500)
	4	5	3
2	(600,-400)	(100,-100)	(-400,200)
	9	7	4
3	(-200,-700)	(-300,-50)	(-400,600)
	10	6	1
4	(-20,800)	(330,400)	(680,0)
	2	3	8
5	(-300,-200)	(-200,200)	(-100,600)
	6	7	5

Table 1. Results of point in polygon analysis for *tPoints* for three instances (for each point in each instance, its coordinate and number of polygon where the point lies in it is specified)

7. CONCLUSIONS AND FUTURE WORKS

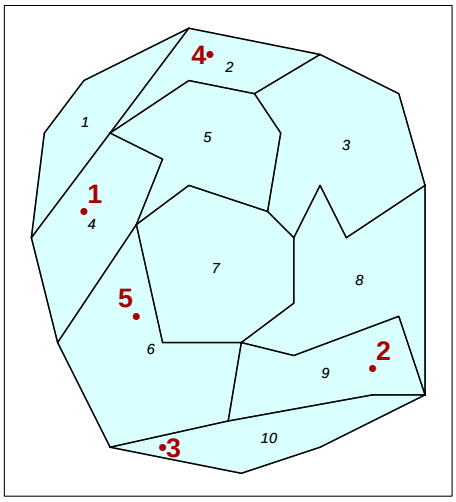
Handling time in GISs is proposed as an essential advancement to GI science and technology. GI theory recommends approaches based on mathematics and computer sciences. What is carried out in this research is testing the hypothesis using the concepts of category theory for integrating non-temporal and temporal point in polygon process. The achieved results and implementation certified this. However, still some questions remain about the level of complexities that would arise in more complex processes.

Using the formalization of time lifting provided by GI theory and the high level of abstraction of functional programming languages enabled us to implement the desired algorithm effectively. The developed codes in this paper are about ten times shorter that their similar codes in other programming environments and they are more comprehensible too. In addition, the implementation of functors and considered reduction of round off error have been achieved using such a programming language.

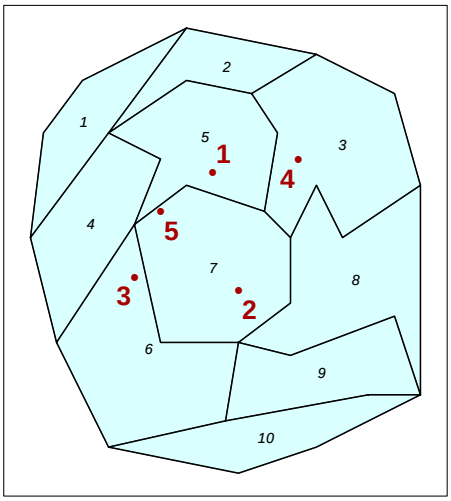
Time lifting that is a kind of functors has been used for including time in GIS analysis with less change in static version of them. Prerequisite of this discussion is algebraic view to GIS. In this way, using higher order languages such as Haskell that can interact with functional variables is essential.

The sample that has been represented in this paper was point in polygon analysis for moving objects.

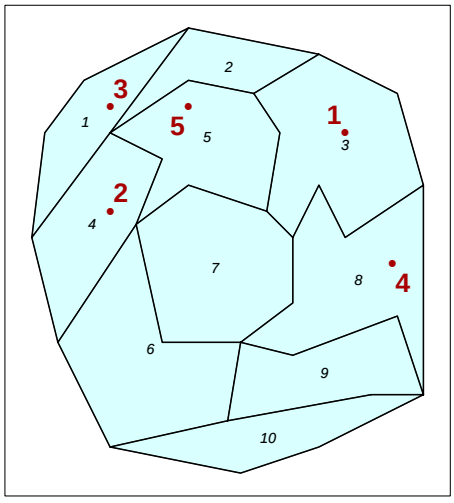
Using this concept for more complicated analysis and also integration of these dynamic analyses with other applications that need dynamic analyses as prerequisite are considered for future works.



(a)



(b)



(c)

Figure 3. Results of point in polygon analysis for *tPoints* for (a) *PinP* 0 (b) *PinP* 10 (c) *PinP* 20

REFERENCES

Berg, M.D., M.V. Kreveld, M. Overmars and O. Schwarzkopf, 2000. "Point Location." In *Computational Geometry: Algorithms and Applications*, Second Edition, Berlin: Springer-Verlag, pp: 121-146.

Burrough, P., and R. McDonnell, 1998. *Principles of Geographical Information Systems*, Edited by P. Burrough, M. Goodchild, R. McDonnell, P. Switzer and M. Worboys, *Spatial Information Systems*. Oxford: Oxford University Press.

Egenhofer, M.J., and D.M. Mark, 1995. *Naïve Geography*, National Center for Geographic Information and Analysis.

Frank, A. U. and A. Gruenbacher, 2001. *Temporal Data: 2nd Order Concepts Lead to an Algebra for Spatio-Temporal Objects*. Paper presented at the Complex Reasoning on Geographical Data, Cyprus, December 1st 2001.

Frank, A. U., 2003. "Ontology for Spatio-Temporal Databases." In *Spatiotemporal Databases: The Chorochronos Approach*, edited by Manolis Koubarakis, Timos Sellis and et al., Berlin: Springer-Verlag, pp: 9-78.

Frank, A. U., *Practical Geometry - Mathematics for Geographic Information Systems*, Oxford: Oxford University Press, submitted 2005.

Grenon, P., and B. Smith, 2004. *Snap and span: towards dynamic spatial ontology*, Spatial Cognition and Computation, 4, no.1, pp: 137-171.

Haskell website: www.haskell.org. (accessed May 2005)

Herring, J., M.J. Egenhofer and A.U. Frank, 1990. *Using Category Theory to Model GIS Applications*, Paper presented at the 4th International Symposium on Spatial Data Handling, Zurich, Switzerland.

Karimipour, F, M.R. Delavar and A.U. Frank, 2005. *Applications of Category Theory for Dynamic GIS Analyses*. Accepted to be published in Proc. GIS Planet 2005, Estoril, Portugal, May 30 - June 2.

Ryu, K.H., and Y.A. Ahn, 2001. *Application of Moving Objects and Spatiotemporal Reasoning*. A TIMECENTER Technical Report.