Applications of Category Theory for Dynamic GIS Analyses

An Algebraic Approach for Achievement of Dynamic GIS Analyses

Farid KARIMIPOUR, Mahmoud R. DELAVAR and Andrew U. FRANK

Geospatial Information Systems (GISs) are widely used to support spatially related decisions. However, recent developments in time-dependent data from historical to real time data capture, force GISs to be extended to handle changes over time. Therefore, existing analysis functions have to be lifted from static to dynamic. Most of past efforts in this area have used computational techniques for special purposes. Therefore, using a general concept to establish dynamic capabilities to a wide range of GIS analyses is useful. This is possible through considering GIS elements and analyses as algebraic structures using category theory and its related concepts. It is intended in this paper to define a functor to be used to transform static GIS analysis functions to dynamic ones using higher order (functional) languages. To achieve this objective, some major analysis functions are investigated in this paper and the results will be generalized to the rest, as further steps of this research.

KEYWORDS

GIS, Dynamic GIS analyses, Spatio-temporal GIS, Algebraic structure, Category theory, Functor, Lifting, Functional programming languages

INTRODUCTION

Geospatial Information Systems (GISs) are widely used to support the spatially related decisions and provide an integrated and flexible set of tools for analyzing large volume of geospatial data. To achieve full GISs' functionalities, they must have capabilities to store and retrieve data (data management), manipulate data (data analysis), and represent data (data visualization).

In many applications, GISs inherent static nature of cartographic mapping and tend to show a static situation of the world that is called a snapshot [3]. In other words, GIS was most often used in an exploratory mode. However, recent developments in time-dependent data from historical to real time data capture, such as environmental, transportation, health, administration, and defense, will lead to many new potential applications for spatio-temporal systems [17]. Enhancing the spatio-temporal capabilities of geospatial information systems is an issue that has received much attention recently and it is led to the development of a GIS branch called spatio-temporal GIS. These information systems manage not only spatial and aspatial components of geospatial entities, but also consider the temporal characteristics of them.

Spatio-temporal GIS components can be obtained from extension of common GIS components. These components are temporal data management, analysis, and visualization [12]. It is obvious that spatio-temporal analyses are based on existing spatio-temporal databases and therefore, most of past efforts to add temporal capabilities to geospatial information systems have been undertaken to spatio-temporal databases and modeling. Time-stamping and event or process based spatio-temporal data modeling are some of the results in this area [8]. On the other hand, temporal visualization is not a pure geospatial information (GI) related problem and the results of other data visualization and representation efforts, such as animation, 3D, and multimedia representations can also be used for representation of spatio-temporal data. A limited number of researches on development of spatio-temporal analyses have been undertaken so far. Many of the GIS analyses, such as optimization and

resource allocation (e.g., Voronoi diagrams) can be more useful, if they incorporate dynamic capabilities. Recently, some efforts have been done in this area [11]. However, most of them have used computational techniques for special purposes and their results can not be generalized for other analyses. Therefore, an important question arises, namely will it be possible to use a general concept to establish dynamic capabilities to a wide range of GIS analyses? The proposed solution could be dealing with an algebraic treatment of different models of GIS. Category theory and its related concepts seem to be a relevant candidate for this end. In this relation, implementation of these concepts is an important challenge and it can be solved using a higher order (functional) programming language.

Section 2 reviews the basic mathematical concepts of this work. In this Section, algebraic structures, category theory, and functors are introduced. In Section 3, the approach for lifting a static analysis to a dynamic one through integrating these concepts is represented. Section 4 contains the definition and specifications of a functional programming language that provides a framework for implementation of the proposed idea and in this relation, Haskell, as a functional programming language, is introduced. In Section 5, the mentioned approach is used for dynamic construction of calculation of area of a changing polygon as a simple and practical GIS analysis. Finally, Section 6 includes conclusions and future works.

2. BASIC MATHEMATICAL CONCEPTS

Proposed approach for lifting static GIS analyses to dynamic ones uses some basic mathematical concepts including algebraic structures, category theory, and functors that are discussed in this section.

2.1. Algebraic Structures

An algebra describes an abstract class of objects and their behavior [6]. In other words, an algebra consists of a collection of types, operations, and axioms. A simple example of an algebra is natural numbers with plus, minus, negate, and zero operations and commutative, associative, existence of identity and existence of inverse as axioms as follows [4]:

Types:	numbers	
Operations:	add (+) , minus (-) , negate, zero (0)	
Axioms:	a + b = b + a	*commutative*
	(a + b) + c = a + (b + c) = a + b + c	*associative*
	0 + a = a + 0 = a	*existence of identity*
	a + (negate a) = 0	*existence of inverse*

Structure of operations in an algebra is independent of an implementation. Thus, behavior of many things can be described with the same algebra, if their behavior is structurally equivalent. For example, the operations with counts and operations that apply to the result of the counting are the same, independent of what is being counted [2].

Algebraic structures seem operable in the GIS applications to simplify thinking about them [2]. For example, although vector and raster GIS has different primitive elements and operations, they have the same algebraic structure. Moreover, real world and our models of real world in GIS, representing a system that is part of reality, are in the same algebraic structures. This property enables us to construct a relationship between them in a special way to move from one space to another.

2.2. Category Theory

Among many algebraic structures, categories are the most operable to GIS applications. 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 [7]. In other words, a category C is to be a class of objects together with, for each pair of objects X and Y in C, a set h(X, Y) whose elements are called maps, such that composition of maps is possible and these compositions are associative [9] (Figure 1). In mathematical language, a category with its elements can be shown as below:

- $\forall A \in C \quad \exists e_A : A \to A \quad \ni [\forall f : A \to B, g : B \to C \Rightarrow e_A.f = f, g.e_A = g]$
- $\forall f : A \to B, g : B \to C \quad \exists h : A \to C \quad \ni h = f.g$
- $\forall f : A \to B, g : B \to C, h : A \to C \Rightarrow (f.g).h = f.(g.h) = f.g.h$

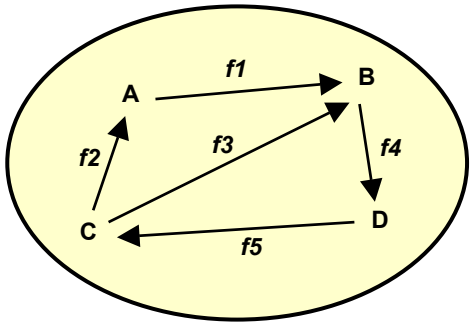

Figure 1: A category with its elements

A categorical viewpoint demonstrates that semantics of operations are independent of the representations they are applied to [2]. 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 [1].

2.3. Functors

Regarding the concept of categories, a morphism or more properly a functor between two categories associates elements and operations from one category to another that preserves the structure and operator algebra [7] (Figure 2). If the first and the second categories are called P and Q, respectively:

- $F(e_A) = e_{F(A)}$
- $\forall f : A \to B \in P, g : B \to C \in Q \Rightarrow$

  $[F(f) : F(A) \to F(B) \in Q, F(g) : F(B) \to F(C) \in Q \quad \ni F(f.g) = F(f).F(g)$.

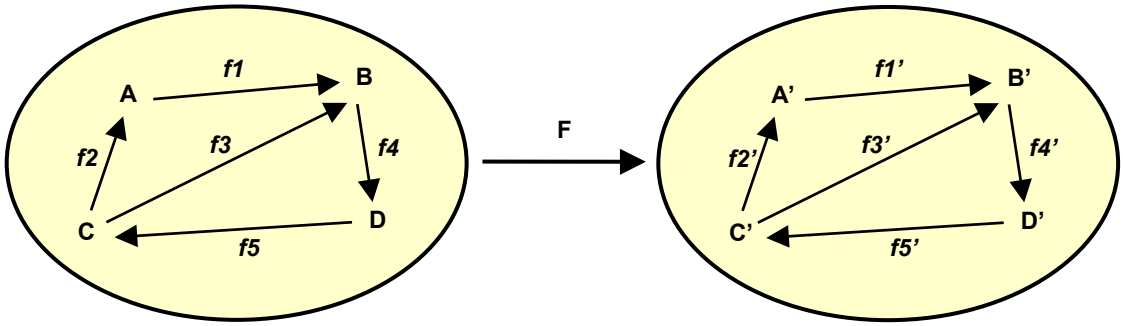

Figure 2: Functor F transforms the first category elements to their associated in the second one

Such relations are commonplace in GIS. For example real world and a defined model of it, can be considered as two categories with the same structure and therefore it is possible to define a functor to associate them together. In GIS, such mappings that preserve algebraic structure are well-known as homomorphism.

3. CATEGORY THEORY FOR DYNAMIC CONSTRUCTION OF GIS ANALYSES

As mentioned before, GIS analyses can be more useful, if they support the time dimension. Some of these analyses can be modified to apply to dynamic data with computational techniques [11]. However, they have two main limitations:

- Most of them consider time as a discrete property, while there are some applications that need a continuous time.
- Because of using computational techniques to modify them, a technique that is used for dynamic construction of one analysis, may not be used for another.

Regarding these limitations, development of a framework for dynamic construction of a wide range of GIS analyses using the same way will be very useful. Indeed, one is in need of a machine that converts a static GIS analysis to a dynamic one.

On the other hand, static and dynamic GIS spaces have the same internal structure. The only difference in these two spaces is in the dimension of their objects. Since category theory concentrates on the processes instead of properties of objects, it seems an appropriate candidate to associate static and dynamic GIS spaces and their objects and processes.

As shown in Figure 3, static and dynamic spaces are two categories with the same structure. Elements of the first category can be divided into three components. The first are primitive objects such as point, line, and polygon. The second are (basic) operators that can not be decomposed to more detailed ones, such as plus (+) and minus (-). The third are functions that are combinations of operators or other functions to do more complex tasks, such as distance measurement and they are the final results of a dynamic analysis. Indeed, in the previous efforts, these functions lift to dynamic situation directly (e.g., [11]). However, if we look at it from a categorical viewpoint, it is sufficient to lift the primitive objects and basic operators and then the functions will be lifted automatically. This achievement has two main advantages:

- In spite of the existence of many functions and complex analyses in GIS, there are finite primitive elements and basic operators in it. Therefore it is possible to lift finite objects (that are primitive elements and basic operators) and then lift many functions and analyses automatically.

- Lifting all primitive objects and basic operators is done in the same manner and this is possible through defining a functor for lifting all objects.

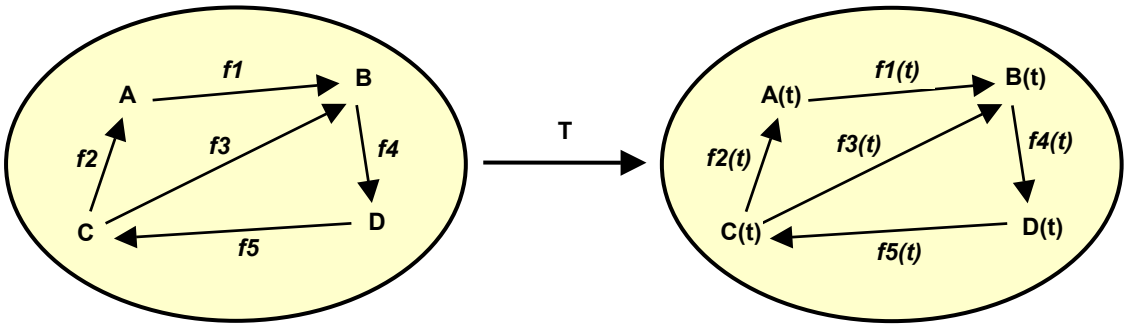

Figure 3: Static and dynamic GIS space category and their related functor (functor T)

Regarding the above mentioned definition of functors, they are functions that have other functions as argument and it is obvious that such functions can not be achieved using common programming languages such as c++. Implementation of these ideas is possible using a higher order (functional) programming language. In the next section, concepts of these programming languages will be reviewed and Haskell, as a sample of such programming languages, will be introduced.

4. HIGHER ORDER (FUNCTIONAL) PROGRAMMING LANGUAGES

Programming languages are classified by orders that are the type of their variable symbols. A zero order language has no variables (only constant). A first order language has variables, which stand for objects, but not for functions. A second order language has variables that can stand for objects or functions [2].

Most previous research in spatial information theory is carried out using first order languages. However, some of the new applications need a higher order language [5]. Even most of the previous GIS analyses can be more easily implemented using such languages. As an example, map algebra of Tomlin [14, 15, 16] has implicitly the concept of higher order functions [5].

Obviously, the proposed treatment for dynamic construction of static analyses of GIS also needs a higher order language. This is the case because functors can operate on any object that is a function. In this case, a functor is a function that takes functions as arguments.

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. They are as old as Fortran [5]. The functional languages, which are strictly typed, use a type system in which functions have proper type and type checking includes the passing of functions as arguments [10].

Haskell is one of the functional programming languages that support our requirements. It has evolved significantly since its birth in 1987. Here, the syntax of Haskell 98 is used. More details about Haskell can be found in [13] and [18].

5. IMPLEMENTATION

In this section, a small category is constructed for a few simple static primitive elements, operators and analyses in GIS and lifted them to a dynamic one using functors.

Time-dependent data in GISs can be divided to temporal location (change in position of objects in time), temporal attribute (change in descriptive properties of objects in time) and combination of them (change in both position and attributes of objects in time). Here, we concentrate on objects that only change their position in time. Indeed, the movement of objects and use of category theory concept and functors to advance the functions upon them to support temporal aspects of data is considered. Because movement can be abstracted to movement of point objects, with no loss of generality, moving points are concentrated in this paper.

Figure 4 shows a category with the following content:

- Primitive elements: Number and Point data type
- Basic operators: plus (+), minus (-), multiplication (*), division (/) (+ and – for both numbers and points; * and / only for numbers), getX, getY, pointCreation, dX ans dY (for Points). It is notable that plus and minus operations operate for both number and point datatypes through using a polymorphism mechanism.
- Functions and analyses: distance and area

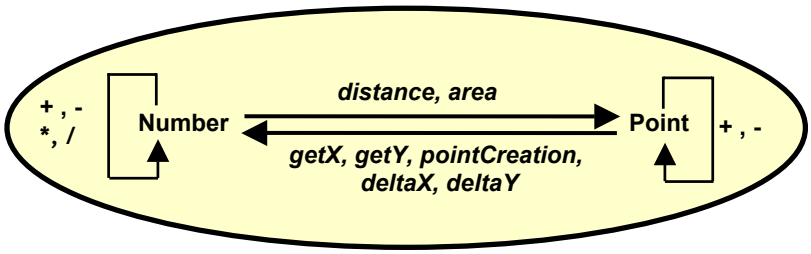

Figure 4: Static category for a few GIS elements and analyses

Data type “Number” already exists and data type “Point” is constructed as follows:

```
data Point a  = Pt a a
```

Basic operators and functions for “Point” are defined as follows:

```
class Points p s where
  getX, getY :: p s -> s
  pointCreation :: s -> s -> p s
  deltaX, deltaY :: p s -> p s -> s
  distance :: p s -> p s -> s
  area :: [p s] -> s

instance Points Point a where
  getX (Pt x1 y1) = x1
  getY (Pt x1 y1) = y1
  pointCreation x1 y1 = Pt x1 y1
```

```
dX p1 p2 = getX p1 - getX p2
dY p1 p2 = getX p1 - getY p2

distance p1 p2 = (dx*dx + dy*dy) where
  dx = deltaX p1 p2
  dy = deltaY p1 p2

area ps = (sum parts) / (one + one) where
  parts = zipWith (*) (zipWith (-) l1 l2) (zipWith (+) l3 l4)
  l1 = map getX ps
  l2 = tail l1 ++ [head l1]
  l3 = map getY ps
  l4 = tail l3 ++ [head l3]

(+) (Pt x1 y2) (Pt x2 y2) = pointCreation (x1 + x2) (y1 + y2)
(-)  (Pt x1 y2) (Pt x2 y2) = pointCreation (x1 - x2) (y1 - y2)
one = (1%1)                    -- “Ratio Integer” type of 1
zero = (0%1)                   -- “Ratio Integer” type of 0
```

Now, to make a similar category for moving points (Figure 5), we should lift the above elements, operators, and functions for dynamic points. As mentioned before, it is sufficient to lift only primitive elements and basic operators. To lift the primitive elements, we define type “Changing” that changes each constant input *v* to a function from a Ratio Integer number to *v*:

```
type Changing v = (Ratio Integer) -> v.
```

Then, “Changing Number” and “Changing Point” are the dynamic versions of “Number” and “Point”, respectively.

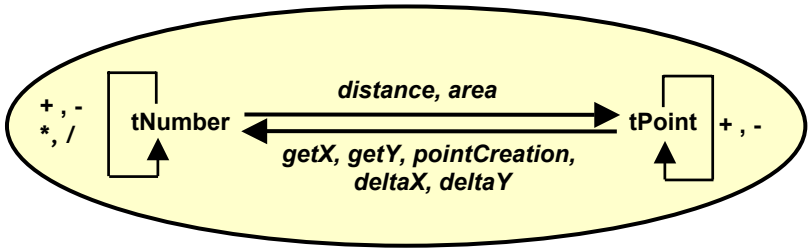

Figure 5: Dynamic version of category illustrated in Figure 4

To lift the basic operators, a functor is needed. We define functors lift0, lift1, and lift2 to lift operators with zero, one, and two parameters, respectively. These functors add a parameter “t” to input functions. Lifting for operators with more arguments can be done in a similar way.

```
lift0 :: b -> a b
lift1 :: (b -> c) -> a b -> a c
lift2 :: (b -> c -> d) -> a b -> a c -> a d

lift0 a = \t -> a
lift1 op a = \t -> op (a t)
lift2 op a b = \t -> op (a t) (b t)
```

Now, to lift each basic operator, it is sufficient to operate functor ‘lift’ to it.

```
(+) = lift2 (+)
(-) = lift2 (-)
(*) = lift2 (*)
(/) = lift2 (/)
zero = lift1 (zero)
one = lift1 (one)
```

Other operators such as getX, getY, pointCreation, deltaX and deltaY, are independent on type of point and can be used for both static and dynamic points. In this way, distance and area analyses, that are the combinations of the basic operators, will lift automatically.

Now, if we have static points, it is possible to use the static elements, operators, and functions to interact with them:

```
p1, p2, p3, p4, p5 :: Point (Ratio Integer)          -- static points
p1 = pointCreation (-1.5) (1.5)
p2 = pointCreation (6.0)  (8.2)
p3 = pointCreation (7.3)  (4.4)
p4 = pointCreation (-2.1) (4.0)
p5 = pointCreation (4.1)  (2.9)

p6 = p1 + p2                              -- static addition of p1 and p2
d12 = distance p1 p2                      -- static distance between p1 and p2
staticPolygon = [p1, p2, p3, p4, p5]      -- static polygon of points p1 to p5
staticArea = area staticPoints            -- static area of static polygon.
```

To have dynamic points, the lifted elements, operators and functions of dynamic category can be used as follow:

```
tp1, tp2, tp3, tp4, tp5:: Point (Changing (Ratio Integer))      -- dynamic points

tp1 = Pt (1) (\t -> 1.1 - 5.4 * t* t)       (\t -> 4.0 - 1.9 * t * t)
tp2 = Pt (2) (\t -> 2.2 * t * t + 0.5 * t) (\t -> 3.0 - 2.3 * t * t)
tp3 = Pt (3) (getX (tp1 + tp2))           (getY (tp1 + tp5))
tp4 = Pt (4) (\t -> 7.0 * t * t + 3.5  )    (\t -> 3.0 + 5.3 * t * t)
tp5 = Pt (5) (\t -> 2.1 - 5.2 * t * t)       (\t -> 4.9 * t * t + 2)

tp6 = tp1 + tp2                                  -- dynamic addition of tp1 and tp2
td12 = distance tp1 tp2                          -- dynamic distance between tp1 and tp2
dynamicPolygon = [tp1, tp2, tp3, tp4, tp5]       -- dynamic polygon of points tp1 to tp5
dynamicArea = area dynamicPoints                 -- dynamic area of dynamic polygon
```

All of the above mentioned left variables are functions with one parameter (time) and it is possible to calculate each of them for an instance, for example “dynamicArea 3.3” represents the area of dynamic polygon for instance 3.3. Figure 6 and Table 1 shows the dynamicpolygon in some instances and results of dynamicArea function for these instances, respectively.

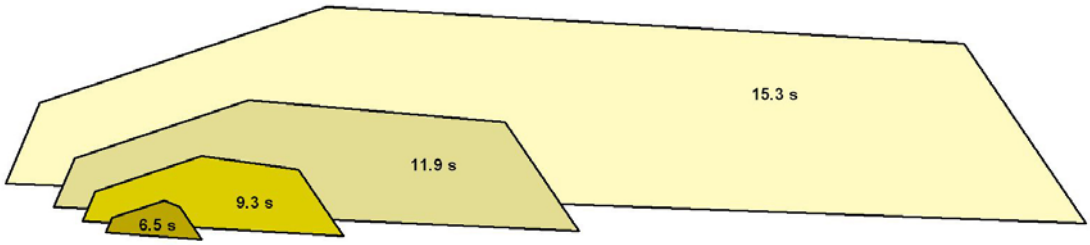

Figure 6: DynamicPolygon in some instances

Instance (Sec.)	Area (Sq. Meters)
6.5	67867.721875
9.3	285245.613879
11.9	765107.612209
15.3	2090772.660699

Table 1: Results of dynamicArea function for dynamicPolygons illustrated in Figure 6

"Ratio" type is used for points. In this type, each number is defined as ratio of two numbers. One of the advantages of Haskell is that it can be delivered the result of calculations of "Ratio" type as "Ratio" type, too. Therefore the results do not have rounding error. The numerical precision of the output can be introduced by the user according to his/her needs.

6. CONCLUSION AND FUTURE WORKS

Including time in GISs is an essential advancement to keep efficiency and usability for spatio-temporal modeling and therefore dynamic construction of analyses of GISs is an important challenge to achieve this end. Regarding wide range of available analyses of GISs and their differences, treatments that have ability to make dynamic versions of all analyses in a similar way seem very useful. Category theory and its related concepts as an opportunity in this direction have been introduced in this paper. 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 category that has been represented in this paper was dynamic area calculation of a changing polygon. Definition of categories with more elements, operators, and functions and also integration of these dynamic analyses with other applications that needs dynamic analyses as prerequisite are considered for future works.

REFERENCES

1. **Alber, R. et al**, Spatial Organization—The Geographer's View of the World, Prentice Hall, USA, 1971.

2. **Frank, A. U.**, A Theory for Geographic Information Systems, unpublished manuscript, 2005.

3. **Frank, A. U. and Gruenbacher**, A., Temporal Data: 2nd Order Concepts Lead to an Algebra for Spatio-Temporal Objects, in Proc. Workshop on Complex Reasoning on Geographical Data, Cyprus, December 1, 2001.

4. **Frank, A. U.**, One Step Up the Abstraction Ladder: Combining Algebras—From Functional Pieces to a Whole. International Conference COSIT'99, Stade, Germany, 1999.

5. **Frank, A. U.**, Higher Order Functions Necessary for Spatial Theory Development, in Proc. Auto-Carto 13, Vol. 5, Seattle, Wash., April 7-10, 1997, pp. 11-22.

6. **Guttag, J. V. and Horning J. J.**, The Algebraic Specification of Abstract Data Types, Acta Informatica 10(1): 27-52, 1978.

7. **Herring J. R. et al**, Using Category Theory to Model GIS Applications, in Proc. 4th International Symposium on Spatial Data Handling, Vol. II, Zurich, 1990.

8. **Langran G.**, Time in Geographic Information Systems, Taylor and Francis, England, 1992.

9. **Mac Lane, S.**, Categories for the Working Mathematician, Springer, Germany, 1971.

10. **Milner, R.**, A Theory of Type Polymorphism in Programming, Journal of Computer and System Sciences, 17: pp. 348-375, 1978.

11. **Mostafavi, M. A. et al**, Dynamic Voronoi/Delaunay Methods and Applications, Journal of Computer and Geosciences, 20: pp. 523-530, 2003.

12. **Nadi, S**, Spatio-Temporal Data modeling in Geospatial Information Systems (GIS), MSc. Thesis, University of Tehran, 2004.

13. **Thompson, S.**, Haskell: The Craft of Functional Programming, second edition, Addison Wesley, England, 1999.

14. **Tomlin, C. D.**, Digital Cartographic Modeling Techniques in Environmental Planning, Ph.D. Thesis, Yale University, 1983.

15. **Tomlin, C. D.**, A Map Algebra, in Proc. Harvard Computer Graphics Conference, Cambridge, Mass, 1983.

16. **Tomlin, C. D.**, Geographic Information System and Cartographic Modeling, Prentice Hall, USA, 1989.

17. **Worboys, M. and Duckham, M.**, GIS: A Computing Perspective, second edition, CRC, USA, 2005.

18. Haskell website: https://www.haskell.org.

AUTHORS INFORMATION

Farid KARIMIPOUR
fkarimipr@ut.ac.ir

University of Tehran

Mahmoud R. DELAVAR
mdelavar@ut.ac.ir

University of Tehran

Andrew U. FRANK
frank@geoinfo.tuwien.ac.at

Technical University of Wien