TEMPORAL TOPOLOGICAL RELATIONSHIPS OF CONVEX SPACES IN SPACE SYNTAX THEORY

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

[a] Dept. of Surveying and Geomatic Eng., Eng. Faculty, University of Tehran, Tehran, Iran, - (rezayan, karimipr, mdelavar)@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: Space Syntax theory, GI science, GI theory, Category theory, and Topology.

ABSTRACT:

GI science development has to be served by effective GI theory. Development of GI theory requires clearer characterisation of GI domain which could illustrate real world better and provide a framework for delineation of rational hypotheses. This aim is followed in recent researches by using mathematics as the basis of GI theory. Axiomatic set theory and formal logic is the foundation of this mathematical treatment for study of structures, changes, and spaces in GI domain. While the potential hypotheses require to be modelled into logical structures and be prepared to be evaluated, computer science is adopted as another foundation of GI theory. One of the recent trends in GI theory development is characterizing the real world as functions and utilization of category theory and algebras as the mathematical basis for handling realities and developing hypotheses. Considering this, development of unique and integrated basis for handling static and dynamic GI concepts is one of the hypotheses which are studied in some researches. Their outcomes define theoretical feasibility of defining morphisms between static and dynamic domains known as functors or time liftings. This approach is evaluated for some models and implemented using functional programming languages, however, evaluations are still required for more different characterization of the world. This paper provided one of these evaluations over a topological characterization of convex spaces in real world described by Space Syntax theory. This theory illustrates human settlements and societies as a strongly connected space-time relational system between convex spaces. Such a system is represented by a connectivity graph. Some morphologic analyses are also defined for deriving properties of the graph that illustrate how the space and time are overcome by relational systems and convex spaces. Investigation of temporality in Space Syntax theory resulted that more dynamicity exists in local scales among activities. Then the specific problem of this paper is defined as modelling integrated static and dynamic analyses of an activity / point based problem in local scale in regards with studying how effective they overcome space and time. The derived model is implemented using a functional programming language known as Haskell. While the most important aim of the paper as validity of the time lifting approach for topological models of convex spaces is obtained, some questions about mixed usage static and dynamic data and how the required computation time and memory is increased are formed.

1. INTRODUCTION

Is it right that over 80 percent of information has spatial factors? Emergence of space's roles as some of the foundational ones in different sciences strengthens more the rightness of this claim. For example in architecture science, theories like Space Syntax theory is developed which illustrates the importance of constructive roles of space in creating societies and proposes that the social construction of space in human settlements is mediated by spatial laws. Then it would be questioned that are spatial laws, which derived out in different sciences, consistent with each other and could these laws be integrated together?

Geographic information theory is served to provide a foundation to support derivation of consistent and integrable spatial laws and theories. GI scientists are developing GI theory through formalistic utilization of mathematics for studying structures, changes, and space. This mathematical trend is generally based on axiomatic set theory and formal logic. It is realized by functionalists through adopting category theory and algebras. This trend is also accompanied with addressing the derived concepts of GI theory in computer science.

Then it would be supposed truly hat GI concepts which have definite mathematical and algebraic structures (like topology) could take advantage of GI theory more, while other GI concepts need to be redefined first. Time is one of the critical concepts have to be redefined in GI theory. Time is inherently linked to space (Egenhofer and Mark, 1995), then, provision of an integrated basis for dealing with static and dynamic concepts is inevitable.

This paper is using the results obtained from formalization of time in GI theory by Frank and his colleagues for evaluation of time integration in spatio-temporal concepts of Space Syntax theory. These concepts are created from investigation of topological relationship of convex spaces in human settlements.

Besides, the definitions illustrated in this paper are implemented into a functional programming language known as Haskell. These are discriminated from other definitions by adding a " > " symbol to their beginnings.

This paper is composed of 9 sections. In section 2 Space Syntax theory is reviewed, the topological properties of space syntax theory and their morphologic analyses are described in section 3, and then time and temporality in Space Syntax theory are

* Corresponding author.

investigated in section 4. In section 5, formalization of time in GI theory is described. Section 6 provides a more specific problem definition which is implemented in section 7 and is followed by a case study in section 8. Finally the results and conclusions are provided in section 9.

2. SPACE SYNTAX THEORY

Could our cities be designed according to scientific and rational laws? Urban design is the process of giving physical design direction to urban growth, conservation and change. It sits at the interface between architecture and planning. While architecture and planning focus on artistic and socio-economic factors, designing emphasis on physical attributes usually restrict its scale of operation to arrangements of streets, buildings, and landscapes (Batty et al., 1998). Architecture, design, and planning are suffered from the lack of scientific theories, as existing theories are mostly normative and weakly analytical.

Space Syntax theory is a spatial theory which attempts to overcome the mentioned theoretical deficiencies by providing the means through which we could understand human settlements. This theory is originally illustrated by Hillier and Hanson (1984) and being used to explore, predict and evaluate the likely effects of design alternatives. It is especially a theory and method for description of invariants in built space.

Space is a container of relations and interactions (Couclelis ,1992 cited in Jiang et al., 2001). Then space is a configurational entity. Space Syntax theory adopts the concept of spatial configuration as its foundation (Hillier, 1996) and use it as the required basis for abstraction and integration of general properties, structures, and transformations in societies and human settlements.

One of the central concepts of Space Syntax theory is urban grid. It is the pattern of public space linking the buildings of human settlements (Hillier, 2001). Considering the strong role of urban grids in creating living cities, their relations with movement are usually investigated. Urban grids are defined as static core elements of urban systems strongly influence the long term dynamicity of urban systems and movement, as the strong force that holds the whole urban system together (Hillier, 2001). Then it is resulted that the relational systems of societies are strongly space-time relational systems which their individual relations are space-time relations and events (Hillier and Netto, 2001).

Based on these outcomes, Space Syntax theory provides its organic definition for a society as an evolutionary abstraction imposed on space-time reality. In this society, space is acted as an inverted genotype. It means that the information needed to reproduce cultural patterns of space is found in the spatial configurations themselves as relations / interactions. Individuals who make up such a organic society (e.g. built areas and activities) are clearly well-defined space-time things and the spaces between individuals are filled up or overcome by the space-time relational systems. These imply the movement in the societies (Hillier and Netto, 2001).

Then it is depicted that the social construction of space in human settlements is mediated by spatial laws which are two kinds: those by which different ways of placing buildings gave rise to different spatial configurations (local and conservative); and those through which different spatial configurations created different patterns of co-presence amongst people through their effect on movement (global and generative) (Hillier, 2001). This conclusion conforms to viewpoint of cognitive perception: space could be considered at two scales, large or small scales (Egenhofer and Mark, 1995). Large scale space is beyond human perception and cannot be perceived from a single point; while small-scale space is presumably larger than the human body, but can be perceived from a single vantage point (Jiang et al., 2000). In general, residential and cultural factors, which are variants, dominate local scale and commercial and micro-economic factors, which are invariants, form global scale in Space Syntax theory.

Dealing with invariants, Space Syntax theory expresses that it is feasible to derive a general pattern, known as deformed wheel (Hillier, 2001), in global scale of all societies through analyses of the space-time relational systems. The deformed wheel patterns are firstly used for explanation of movement. Also the effect of variants on society are analyzed studying the level of deformation occurred in the deformed wheel patterns. Moreover, some results about the usages (residential and commercial), density, homogeneity and complexities of patterns in societies are derived from movement analyses.

In short, Space Syntax theory defines the relation of space and society as a two way generic and systematic relation (Hillier, 2001). Cities are defined as a transformation of space-time and a transformation of society (Hillier and Netto, 2001). This theory generates topological formal models of space-time relational systems of cities. These systems are represented as connectivity graphs and also equipped with effective methods for analyzing their morphologic properties (Section 3).

3. TOPOLOGICAL RELATIONSHIP OF CONVEX SPACES IN SPACE SYNTAX THEORY

The mentioned graph representations of societies' relational systems are generated through the following procedure:

1. Spatial decomposition of spatial configuration into elementary units of analysis: bounded spaces, convex spaces and axial lines. The analysis units are defined as follow (Brown, 2001):
   - Bounded spaces (typical enclosable rooms with doors) usually correspond to functional use designations (Figure 1).
   - Convex spaces identify the extent of spatial decomposition and usually correspond with privatization and localization of space. (Figure 1.a).
   - Axial lines as straight lines which identify the extent of spatial continuity from the entrance to the system through it and usually correspond with flows and globalization of space. They connect all incidences of the derived convex spaces based on inter-visibility existence between them. (Figure 1.b).

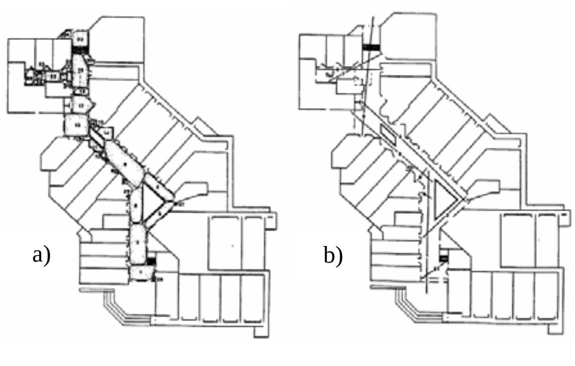

Figure 1. Example of analysis units' extraction of Space Syntax theory (Brown, 2001)

2. Representing these units and their connectivity as a connectivity graph in which its nodes and links are respectively the units and their connections. These graphs are usually big, shallow, non-denderitic, highly integrated, and everywhere ringy (with a large number of cycles) rather than tree-like. This structure can serve the sustainable evolution of societies (Hillier and Netto, 2001). Jiang et. al (2000) illustrates three different representations of a connectivity graph depending on the degree of linearity in the environment. These are:

– Relatively linear / axial representation, where this linear property represents the fact that the built environment is relatively dense, so that the free space is stretched in one orientation at most points (e.g. a city, a town, a village or a neighbourhood) (Figure 2).

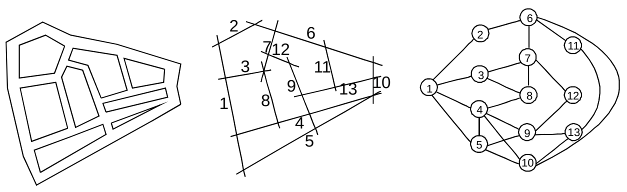

Figure 2. Axial representation (Jiang et al., 2000)

– Non-linear / convex representation, where the free space is partitioned into finite number of convex spaces represented by a convex polygon in 2D maps (e.g. internal layout of a building) (Figure 3).

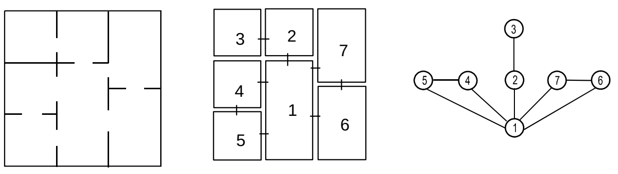

Figure 3. Convex representation (Jiang et al., 2000)

– Non-linear but with more precise spatial presentation / grid representation, where the free space is partitioned into finite number of points and these points visual fields are extracted (Figure 4). Visual field is the space wholly visible from a single vantage point. It is based on the notion of isovist (Benedikt, 1979).

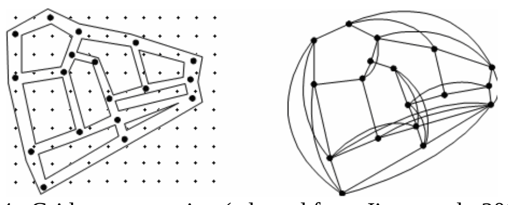
Figure 4. Grid representation (adapted from Jiang et al., 2000)

3. Analyses of the graphs for deriving their morphologic properties: connectivity, control value, depth, integrability, and intelligibility. These are defined as follow (Jiang et. al, 2000, Brown, 2001):

– The connectivity value is defined as the number of immediate neighbours of nodes (1). It is the same as degree of nodes in graph theory.

$$C_i = k \qquad (1)$$

where C_i = Connectivity of ith node
k = Immediate neighbours

– The control value of a node expresses the degree to which the node controls access to its immediate neighbours, taking into account the number of alternative connections of these neighbours. It is defined as sum of the reciprocal connectivity value of nodes directly linked to a node (2).

$$ctrl_i = \sum_{j=1}^{k} \frac{1}{C_j} \qquad (2)$$

where $ctrl_i$ = Control value of ith node
k = Connected nodes to ith node
C_j = Connectivity of jth node

– The depth value is the smallest number of steps from a node to the others. It is defined as total depth and mean depth values (3).

$$D_i = \sum_{j=1}^{n} d_{ij} \qquad (3)$$
$$MD_i = D_i/(n-1)$$

where D_i = Total depth value of ith node
d_{ij} = shortest path between ith and jth node
n = Number of nodes
MD_i = Mean depth value of ith node

– Integration value that indicates the degree to which a node is integrated or segregated from the system. A node is said to be more integrated if all the other nodes can be reached after traversing a small number of intervening nodes and less integrated if the necessary number of intermediate nodes increases. The integration of a node is measured similar to relative asymmetry as the average depth of the node to all other nodes (4).

$$RA_i = 2(MDi – 1)/(n-2) \qquad (4)$$

where RA_i = Relative sssymetry value of ith node

– While the mentioned properties can be measured both locally and globally (especially the integration) the correlation between these two level could describe their part-whole relationship. This property is denoted as intelligibility and defined as the correlation coefficient between local and global properties.

These morphologic analyses are carried out targeting each analysis units or nodes of the graph against the others. This could be interpreted as re-arranging the structure of the graph based on the target node. This process is defined as creating a justified graph (j-graph) for a node. J-graphs are viewpoints of individuals to their society. Justification of a graph is done by putting the target node at the lowest / root position, where it can be distinguished explicitly and from which the whole graph can be seen. The structures of j-graphs are used for visual interpretation of the target nodes properties. While all j-graphs and the main graph of society are homeomorphic, Space Syntax theory concludes that individual and society are different ways of looking at the same thing.

4. TIME IN SPACE SYNTAX THEORY

Temporality in Space Syntax theory emerges from changes caused under the notion of movement which is discussed in section 2. Time here is investigated in two levels of granularity: local and global levels. During the life of a city space, it changes slowly (global scale) and its related activities (interactions and co-presence) change rapidly (local scale).

4.1 Time in Global Scale

City design is a temporal art. However this temporality can rarely be explained and controlled as limited sequences. Moreover the temporal sequence of cities can be reversed, interrupted, abandoned, and cut across (Lynch, 1986). It is similar to temporality in biology (Frank, 2005). Space Syntax theory defines built environments as organic structures too. Therefore, movement at global scale is known as general movement which is highly governed by invariant rules like micro-economy (Hillier, 2001).
Repeating that Space Syntax theory deals with space-time as a genotype, the main spatio-temporal question in global scale would be how a society is re-produced through time by being realized in space (Hillier and Netto, 2001). The abstraction emerges here is due to dominance of knowing which structures are reproduced and overcome space and not dealing with structure itself. Then a society is defined as a continuous space-time entity (Hillier and Netto, 2001).
This notion of temporality and slow changes is adopted with all three representations of environment (Section 3), especially axial and convex representations.

4.2 Time in Local Scale

In local scale, we face with discrete freely mobile individuals who carry out activities. No activity per se generates fast changes and immediate new patterns of space, but they have a certain distribution of demands on co-presence and global movement. When assessing the impact of new activities on space, then, what we need to compare is not so much the contents of new activities but the range of demands they are likely to make on co-presence. (Hiller and Netto, 2001).
The main question of space and time here is how independent activities of large number of individuals / agents in different locations create the overall pattern of cities? Two main specifications of activities are:

1. They are movable and do not accumulate into space-time to create larger forms
2. They are governed by social rules and conventions.

In this scale temporality could be investigated more effectively in the grid representation of environment where grid points could represent positions of activities in space and time. Then the concepts of moving objects could be adopted in Space Syntax theory, too, modelling dynamic activities, their interactions, and co-presences.

5. TIME IN GI SCIENCE

Space and society change each other. While any changes is due to time and change in socio-economic or natural environment, it attracts the attention of societies (esp. the politicians) (Frank, 1998), we should try to effectively formalize and implement time in GIS. Time and space are fundamental and different dimensions of reality in which people live (Frank, 2005; Egenhofer and Mark, 1995).
Despite many efforts and researches carried out in space and time, these two did not advance co-ordinately. Effective handling of time is still a controversial topic in GI science and technology preliminarily. GI scientists enumerate temporal deficiencies as one of the main troublesome issues, besides the defects occurs due to lack of effective integration approaches (Frank, 2005). Also the problems could be detailed as follow:

- Lack of a comprehensive and basic spatio-temporal ontology.
- Considering time as a discrete or partial continuous property of our world while our unique physical reality is governed by differentiable laws.
- Dominance of analytical approaches, suffered from the limitation of computer numbering systems.
- Underestimation of common behaviours of models which are used in GI domain (e.g. network, object and field) that resulted in creation of context-based temporal viewpoints.

The general reason for these deficiencies is the lack of effective GI theory. A GI theory, like theories of other sciences, would consist of a formal language and rules concerning valid (simple) relationships and facts within the language (Frank, 2005). This theory would be broad and comprehensive enough to cover all domains of GI science and technology.
After about one decade of dealing with GI theory development, recently, it is somehow concluded that mathematics could provide the required basis of GI theory. Mathematics is treated in GI theory as study of structures, changes, and space (further than figures and numbers) and used into a formalistic approach, which is based on axiomatic set theory and formal logic. This treatment is accompanied with computer science adoption as the basis for hypotheses implementation and evaluation.
The mentioned trend in GI theory is followed in some of the recent advancements like:

- Development of the basic form ontology for space (SNAP) and time (SPAN) by Grenon and Smith (2004) and their colleagues.
- Development of multi-tier ontology by Frank (2003).
- Development of functional approaches for GI on the basis of category theory by Herring et al. (1991), Frank (2005) and his colleagues.

5.1 Functional Approach in GI Theory

How are the interactions in complex real world carrying out simply? The general answer of functionalists is existence of abstraction interfaces which absorb complexities of real worlds. Then the resulted abstract real world could be interacted extensionally, free from any computation. The entities here would be relations in real world, or more properly functions.

This viewpoint is employed in different sciences, especially social sciences, philosophy, and architecture. In GI science, one of the recent functional approaches for definition of GI related interactions is described by Frank (2003) as the closed loop of semantics. He describes interactions as composition of a series of functions carry out by agents in reality (Figure 4).

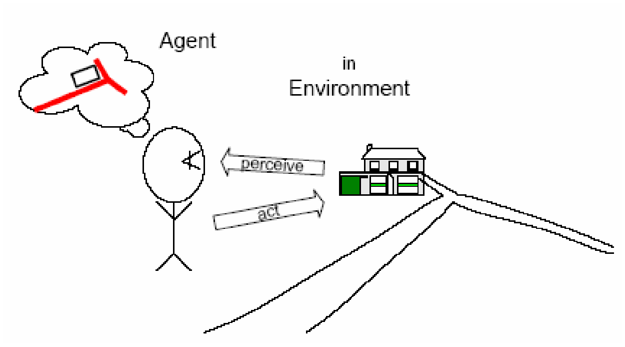

Figure 4. Closed loop of semantics

These functions could be expressed as (5)

$$Reality \rightarrow Observation \rightarrow Modeling \rightarrow Act \rightarrow Reality \qquad (5)$$

,or in their commutative version as (6):

$$\begin{tikzcd} Reality \arrow[r] \arrow[d] & g \arrow[r] & Reality \\ Observation \arrow[r] & g' \arrow[r] & Act \arrow[u] \end{tikzcd} \qquad (6)$$

where g = Processes/models in real world.
g' = Processes/models in agent's mind.

Input to *observation* function and output from *act* function are the relations / functions from real world.
What is denoted here as g' function is a functional representation of a GIS. This function could be defined as composition of a series of functions (7), too, like what is presented for TIGRIS by Herring et al. (1989).

$$g' : External\ Modelling \rightarrow Conceptual\ Modelling \rightarrow Logical\ Modelling \rightarrow Physical\ Modelling \qquad (7)$$

Category theory is introduced as the mathematical basis could support this functional representation of GI interactions. Category theory is a mathematical theory that abstractly (no semantics) deals with mathematical structures and relationships between them. It is an attempt to capture what is commonly found in various classes of related mathematical structures. Fundamental concepts of category theory are categories and functors.
A category is a collection of primitive element types, a set of operations upon those types, and an operator algebra which is capable of expressing the interaction between operators and elements (Herring et al., 1989). Considering category C, element types are objects of category (*ob(C)*), operations are morphisms which preserve structures known as homomorphisms (*hom(C)*), and operator algebra is composition of morphisms. The associativity of morphisms' composition and existence of identity morphism are axioms of a category.
For example a field category consists of fields like $(G, +, *)$ and $(H, ++, **)$ as objects and field homomorphism (8).

$$m : G \rightarrow H \qquad (8)$$
$$m(u + v) = m(u) ++ m(v)$$
$$m(u * v) = m(u) ** m(v)$$

where m = homomorphism function
u and v are in G

A functor is a morphism which associates elements and operations from one category to those another, which preserves the operator algebra (Herring et al., 1989). Functor F from category C to category D (9) associates to each object x in C an object $F(x)$ in D and to each morphism $f : x \rightarrow y$ in C a morphism $F(f) : F(x) \rightarrow F(y)$. It also holds identity and composition properties.
Then a category is itself a type of mathematical structure that its structure is preserved by functors. It means that category theory can be described in itself.
While many mathematical theories attempt to study a particular type of structure just by relating the structure to another simpler and better understood structure, category theory used to take ideas to another simpler or even more complex ones (two-way) by studying the structures and morphisms. For example, our approach moving from static to dynamic structure is also towards more complexity.
By definition of all GI concepts like geometry into formal mathematical structures and also their morphisms, GI theory would be formed as a collection of categories. Then any kinds of transformations and integrations of GI concepts to each other would be possible dealing with morphisms within and among categories.

5.2 Functional Formalization of Time

Considering C_s and C_d categories which carry one kind of mathematical structure in their static and dynamic modes respectively, a functor from the static category to the dynamic one can lift us to temporal domain. This functor will get a function that can be used for static inputs and generate a function could be used for dynamic / temporal inputs. This kind of transformation from static to dynamic domain is usually mentioned as *time lift*. While it is possible to define a unique time lift function, for simplicity, different time lift functions are illustrated here (9) for functions with 0 to 3 parameters using lambda calculus syntax.

```
> lift0 a = \ t → a                                  (9)
> 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)
```

where *lift0, lift1 lift2,* and *lift3* = Time lift functions
op = Input function
a, *b*, and *c* = Inputs for *op* function
t = Time parameter
\ = Lambda symbol

For example time lifting of a two parameter function (10) is done using *lift2* function (11).

$$f : a \rightarrow b \rightarrow c \qquad (10)$$

where a, b, and c = Static types

$$g : (t\rightarrow a) \rightarrow (t\rightarrow b) \rightarrow (t\rightarrow c) \qquad (11)$$
$$g = lift2\ f$$

where $(t\rightarrow a)$ and $(t\rightarrow b)$ = Dynamic input functions
$(t\rightarrow c)$ = Dynamic output functions

<u>example:</u> if $f = (+)$ then
$f\ 1\ 2 = 1 + 2 = 3$
$g\ (t + 1)\ (2t) = 3t + 1$
$g\ 0 = 1;\ g\ 1 = 4;\ g\ 2 = 7$

Then any type of value could be structured as a dynamic value by being defined as a function from time to the value (12)

$$Changing\ v = Time \rightarrow v \qquad (12)$$

where v = Any type
Time = Time parameter

Assignment of similar operators and processes to different types is feasible based on polymorphism. Overloading is defined as ad-hoc polymorphism which is done by instantiating different types for a class.
In regards with definition of static and dynamic types, the overloading is just required for basic functions. Other functions could be defined for both static and dynamic modes polymorphicly, using the overloaded basic functions. This means that by general initialization of overloaded functions, further definition of functions could be used for both static and dynamic types.

5.3 Analytic Issues – Computer Numbering System

As mentioned in section 5, two problems for handling time in GI science are treating time as a discrete entity and dominance of analytical approaches. These are outcomes of existing floating numbering systems of computers bounded-ness. It means that just limited number of digits can be used for integer and floating parts of a number. It causes overflowing and round off errors which subsequently result in continuity problem.
The computing environments supporting lazy evaluation has solved the problem of overflowing by validating definition of infinite series. Lazy evaluation means extensionally computation of statements and just evaluating required parts of the statements (13).

```
> naturals = [1, 2 ..]                                  (13)
> n_5 = take 5 naturals = [1,2,3,4,5]
```

where *naturals* = Infinite series of natural numbers

The theoretical solution for round off error problem, which is provided by Franklin (1984), is utilization of rational numbering system. A rational number is a ratios of two integers usually written as a/b where b is not zero. The set of all rational numbers is denoted by ***Q*** which is a linear, dense subset of real numbers, and totally ordered. Being a dense subset means that between any two rationals sits another one (in fact infinitely many other ones).
So what is the problem of substituting rational field numbering systems for float number systems of computers? Rational numbers are structured types (like records) and the computing environment which supposed to used them have to be able two overload numerical operators and functions for structured data types, too. Knowing that overloading is feasible in object oriented environments, the existing computing environments are imperative or do not support overloading for structured data types, especially overloading of operators (like =, +, and *).
One of the major outcomes of using rational numbering systems would be elimination of error generation and propagation in GI algorithms due to round off error.
By defining a rational type as *Ratio Integer*, its temporal counterpart could be defined (14).

```
> Changing (Ration Integer) = Time → (Ratio Integer)      (14)
```

In continue the following synonyms (15) will be used:

```
> type RI = Ratio Integer                                 (15)
> type CRI = Changing (Ratio Integer)
```

The static and dynamic rational types are overloaded over a field class (16).

```
> class Field a where                                     (16)
>         (+),(-),(*),(/) : a → a → a
```

The overloaded instances are as follow:

```
> instance Field (RI) where                               (17)
>         …
> instance Field (CRI) where
>         (+) = lift2 (+)
>         (-) = lift2 (-)
>         (*) = lift2 (*)
>         (/) = lift2 (/)
```

Then any function, using these basic operators, would be valid for both static and dynamic types (18).

```
> dm : (Field a) => a → a → a                             (18)
> dm a b = (a + b) * (a – b)
```

<u>*example:*</u>

```
>               dm 1/2 2/3
>               dm (1/2t) (2/3t)
```

6. PROBLEM DEFINITION

Considering the mentioned issues in previous sections, the aim of this paper can be restated as implementing unique and integrated analyses for static and dynamic topological relationships of convex spaces illustrated in Space Syntax theory (section 3).
Dealing with this, the grid representation of environment is selected as the basis. It is the simplest representation due to simpler treatment required for deriving convex spaces. As the vantage points are predefined as finite set of points, they all can be proposed as potential incidences of convex spaces and their linkage derived out by applying finite number of inter-visibility relations between them.
Moreover, this environment supports more rapid movement and changes in local scale dealing with activities which place in different grid points at different time (Section 4.2). Then our problem could generally be reduced to dealing with a finite set of static or moving activities positions as points which their topological relations are formed based on their inter-visibility relations within an underlying static environment. This environment consists of static passages (e.g. streets) and barriers (e.g. buildings) which enable or disable movement and inter-visibility of points.
The resulted connectivity graph will consist of nodes as the static or dynamic points and links as their inter-visibility. This graph would be the basis of deriving the morphologic analyses described in section 3 except the intelligibility analysis, which is too complicated to be implemented here.
It is expected that this approach could support the analysis of any kind of dynamic activities in regards with how effective they overcome space and time, for example city public services (like public transportation, safety, and security services).

7. IMPLEMENTATION

Implementation deals with defining the following issues:

1. dynamic activities as dynamic points
2. connectivity graph
3. static environment
4. functions

While the mentioned rational numbering system is used in implementation, any notion of numbers in following sections refers to rational numbers.

7.1 Dynamic Activities / Points

A general point is defined as an algebraic data type (19).

```
> data Point a = Point Id a a                                  (19)
```

where *Point* =Type name and constructor of a point
a = Type of the coordinates
Id = Unique identifier for a point (Number)

Then *Point (RI)* defines a static point type that takes rationals for its coordinates. A dynamic point type with dynamic rational coordinates would be defined as *Point (CRI)*.
Considering that calculation of points' multiplication and division are not required, the basic functions for point data type are defined into a class denoted as *Points* (20).

```
> class Points p s where                                       (20)
>         x, y :: p s → s
>         x (Point _ cx _) = cx
>         y (Point _ _ cy) = cy

>         xy :: s → s → p s
>         xy cx cy = Point (-1) cx cy

>         (+) :: p s → p s → p s
>         (-) :: p s → p s → p s
```

where *Points* = Class name and constructor
xy = Constructs a point from x and y coordinates
(+) and *(-)* = Summation and subtraction of points

While most of the functions of class *points* are general and have defaults definition, just (+) and (-) functions of class *points* are overloaded for static (21) and dynamic (22) point data types.

```
> instance Points Point a where                                (21)
>         (+) (Point _ x1 y1) (Point _ x2 y2) =
>                             Point (-1) (x1 + x2) (y1 + y2)

>         (-) (Point _ x1 y1) (Point _ x2 y2) =
>                             Point (-1) (x1 - x2) (y1 - y2)
```

```
> instance Points Point (Changing a) where                     (22)
>         (+) = lift2 (+)
>         (-) = lift2 (-)
```

where _ = Any value

The other basic operations such as equality and ordering of points could be defined similarly not mentioned here.

7.2 Connectivity Graph

The proposed connectivity graph is a set of binary relations between points (23) (Thompson, 1998).

```
> data Set a = Set [a]
> type Relation a = (a,a)                                      (23)
> type Graph a = Set (Relation a)
```

where *Set* =Type constructor of a set
Relation =Type constructor of a binary relation.
Graph =Type constructor of a graph

Also nodes are defined as an algebraic data type (24).

```
> data Node = Node Id ([Id], C, Ctrl, D, MD, RA)               (24)
```

where *Node* =Type constructor of a node
Id = Unique identifier for a node which is the same as its correspondent point id.
[Id] = List of connected nodes Ids

C, Ctrl, D, MD, and *RA* = Parameters calculated from Space Syntax theory's morphologic analyses as connectivity (C), control (Ctrl), total depth (D), mean depth (MD), and integration (RA). They are all static numbers.

Nodes manipulation functions are defined as some *set* and *get* functions (25).

```
> setNodeId :: Node → RI → Node                                    (25)
> setNodeId (Node id param) id' = Node id' param

> getNodeId :: Node → RI
> getNodeId (Node id _) = id

> getConnections :: Node → [RI]
> getConnections (Node _ (cs, _, _, _, _, _)) = cs

> setConnections :: Node → [RI] → Node
> setConnections (Node id (cs,c,ctrl,td,md,ra)) cs' =
                                       Node id (cs',c,ctrl,td,md,ra)

> getConnectivity, getControl, getTotalDepth,
> getMeanDepth, getIntegrability :: Node → RI
> getConnectivity (Node _ (_, c, _, _, _, _) = c
        …

> setConnectivity, setControl, setTotalDepth,
> setMeanDepth, setIntegrability :: Node → RI → Node
> setConnectivity (Node id (cs,c,ctrl,td,md,ra)) c' =
                                       Node id (cs,c',ctrl,td,md,ra)
        …
```

In continue a graph with static numbers is used which is defined as *Graph RI*.

7.3 Static Environment

As mentioned in section 6 the required static environment would define movement barriers and passages of the environment. This could be modelled as a set of planar polygons which do not intersect or include each other (26). Spaces between these polygons define the passages.

```
> data Environment a = [Polygon a]                                 (26)
```

where *Environment* = Type constructor of an environment

A polygon is defined by its bounding straight line segments named as edges. An edge (27) and a polygon (28) are also defined as algebraic data types.

```
> data Edge a = Edge (Point a) (Point a)                           (27)
```

where *Edge* = Type constructor for an edge

```
> data Polygon a = Polygon [Edge a]                                (28)
```

where *Polygon* =Type constructor for a polygon

In general, passages are modelled as paths which are defined for guiding moving points. A path consists of a set of directed straight lines meet each other at different times (Figure 6).

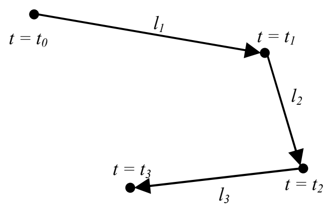

Figure 6. A path consists of multiple directional straight lines at different times

Then a path could be constructed as a conditional statement of multiple directional straight lines based on time (29).

```
> path = cond [c1, c2, … ] [l1, l2, l3, …]                         (29)
```

where *cond = Checks a series of condition and select their respective action*
$c_1 = t_0<t<t_1$
$c_2 = t_1<t<t_2$
l_1, l_2 and *l_3*= Line equations

So the proposed dynamic points / activities would be defined by paths.

7.4 Functions

areIntervisible function is one of the basic functions which checks for existence of inter-visibility relation between two points in an environment (30).

```
> areIntervisible :: (Bools b, Points Point a) =>                  (30)
                    Environment a → Point a → Point a → b
```

where *Bools* = Type constructor of Boolean class which static and dynamic Booleans are overloaded on it

Noting that *areIntervisible* function uses edges intersection process, further definitions are omitted here. *deriveIntervisibles* (31) and *deriveRelations* (32) functions are defined for derivation of all intervisibility relations of a point. Also *mDeriveRelations* (33) is defined for generalizing *deriveRelations* function over all points of a list.

```
> deriveIntervisibles:: (Points Point a) =>                        (31)
      Environment a → [Point a] → Point a → [Point a]
> deriveIntervisibles env ps p = filter (areIntervisible env p) ps

> deriveRelations:: (Points Point a) =>                            (32)
      Environment a → [Point a] → Point a → [Relation RI]
> deriveRelations env ps p =
   [(getID p, getID q) | q ← (deriveIntervisibles env ps p)]

> mDeriveRelations:: (Points Point a) =>                           (33)
      Environment a → [Point a] → [[Relation RI]]
> mDeriveRelations env ps = map (deriveRelations env ps) ps
```

Besides, functions are used for flattening the result of *mDeriveRelations* into one list of relations and also for removing duplicate relation which is not described here.
Then, *cGraph* function (34) is defined to construct the connectivity graph recursively.

```
> cGraph :: (Points Point a) =>                          (34)
            Environment a → [Point a] → Graph RI
> cGraph env ps = Set (mDeriveRelations env ps)
```

Graph nodes are generated from the points list using *makeNodes* function (35).

```
> makeNodes :: (Points Point a) => [Point a] → [Node]    (35)
> makeNodes [] = []
> makeNodes (p:ps) =
                    (Node (pID p) ([],0,0,0,0,0)) : makeNodes ps
```

In continue most of the defined functions are defined in two parts:

1. functions for component analysis
2. functions for generalization of the component analysis functions for list analysis. These functions are defined by prefix *m*

The morphologic analyses of the graph are defined as follow:

1. Connections list:

```
> deriveConnections:: Graph RI → Node → Node             (36)
> deriveConnections g n = [b | (a,b) ← g, a == (getNodeID n)]
                    ++ [a | (a,b) ← g, b == (getNodeID n)]

> mDeriveConnections:: Graph RI → [Node] → [Node]        (37)
> mDeriveConnections g ns = map (deriveConnections g) ns
```

2. Connectivity value:

```
> connectivity :: Node → Node                            (38)
> connectivity = setConnectivity.length.getConnections
```

where *length* = Returns number of elements in a list.

```
> mConnectivity:: [Node]→ [Node]                          (39)
> mConnectivity ns = map pControl ns
```

3. Control value:

```
> control :: Graph RI → Node → Node                      (40)
> control g n = setControl (sum (map reci.getConnectivity
(findNodes g (getConnections n))) n
```

where *findNodes* = Get a list of node ids and returns list of nodes.
reci = Returns reciprocal of a number.
sum = Returns summation of a list of numbers.

```
> mControl :: Graph RI → [Node] → [Node]                 (41)
> mControl g ns = map (pControl g) ns
```

4. Depth value:

```
> depth :: Graph RI → Node → Node → RI                   (42)
```

Readers are referred to Thompson (1998) (pp. 332-334) for definition of *depth* function.

```
> totalDepth :: Graph RI→ Node → RI                      (43)
> totalDepth g p =
     setTotalDepth (sum [depth g p q | q ← graph]) p

> mTotalDepth:: Graph RI → [Node] → [Node]               (44)
> mTotalDepth g ns = map (totalDepth g) ns

> meanDepth :: Graph RI → Node → Node                    (45)
> meanDepth g n =
     setMeanDepth ((getTotalDepth n) / (length g – 1))

> mMeanDepth :: Graph RI → [Node] → [Node]               (46)
> mMeanDepth g ns = map (pMeanDepth g) ns
```

5. Inegrability value:

```
> pIntegrability :: Graph RI → Node → Node               (47)
> pIntegrability g n=
      setIntegrability (2 * (getMeanDepth n - 1) / (length g – 2))

> mIntegrability:: Graph RI → [Node] → [Node]            (48)
> mIntegrability g ns = map (pIntegrability g) ns
```

Then all the morphologic analyses can be composed (49).

```
> calcParam :: Graph RI → [Node] → [Node]                (49)
> calcParam g = ((mIntegrability g).
                 (mMeanDepth g).(mTotalDepth g).
                 (mControl g).mConnectivity).
                 (mDeriveConnections g)
```

As shown, we defined a series of static and dynamic functions which interact with each other. While the basic analyses like deriving intervisibilities are carried out by dynamic functions the functions which analyse graph are static per se. All these functions are composed into a final function (50).

```
> analyseGrid :: (Points Point a) =>                     (50)
        Environment a → [Point a] → Changing (Graph RI)
analyseGrid env ps = ((calcParam g).(makeNodes g))
        where
          g = cGraph env ps
```

example:

f = analyseGrid env1 point1

f 0, f 10 will generate and analyse grid for these times.

8. CASE STUDY

The case study is defined as analysing the quality of overcoming space and time by a public bus transportation system in a city. The city environment is implemented with five static polygons and seven buses as the dynamic activities (Figure 7). Two sample buses' definition are presented in (51).

```
> Activity1 = cond [] [l1_s1]                                  (51)
> l1_s1 = Pt 1 (\ t → 10*t + 100) (\ t → 1440)

> Activity2 = cond [l3_c1] [l3_s1,l3_s2]
> l3_s1 = Pt 3 (\ t → 700) (\ t → 14*t + 600)
> l3_c1 = \ t → t < 50
> l3_s2 = Pt 3 (\ t → 8*t + 300) (\ t → 1320)
```

While path of *Activity1* consists of on straight line, path of *Activity2* has two connected straight lines which are controlled by one condition. These two activities' paths are shown in Figure 7 as directed dashed lines.

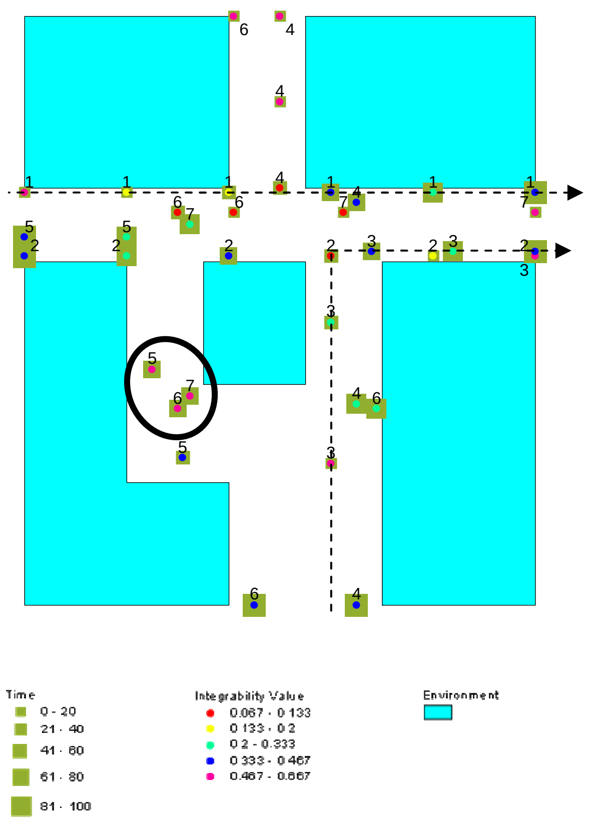

Figure 7. Result of case study execution

The resulted *analyseGraph* function is used for times 0, 25, 50, 75, and 100. The results are presented in Figure 7 as graduate box symbols. Also integrability value is the calculated morphologic parameter selected arbitrarily to be shown over time symbols. A region of high integrability between times 50 and 70 is also shown in Figure 7 by a thick ellipse.

9. CONCLUSIONS

Achievement of the aim followed in this paper fosters validity of current trend of employing functional viewpoint for characterizing of real world and using mathematics for modelling them towards integration of different GI concepts in GI theory. Then it is more likely that integration of static and dynamic domains could be handled following the time lifting approach.
In regards with characterization of real world provided by Space Syntax theory, while its effectiveness for understanding the spatial laws of humankind settlements is illustrated, its conformance with GI concepts and its consistency to be handled and integrated with other spatial theories is tested successfully.
Also the specific approach in modelling dynamic activities and analysing how effective they overcome space and time could be widely used especially for urban related public services. In continue the interaction of different kind of activities could be studied by adding different specific properties of each type of activities as supplementary properties to their connectivity graphs.
Moreover our implementation could ascertain more the suitability of functional programming languages for evaluating GI theory concepts. Succinct, comprehensible, and testable functions of these languages provide us the required framework for simpler and more efficient implementation and evaluation of hypotheses.
Also some considerations about mixed used of static and dynamic analysis and more clarification of computation time and memory growth during time lifting is required. While functional programming environments try to manage memory automatically, more specific techniques for controlling memory allocations and garbage collections are required. All these imply utilization of more sophisticated functional programming language compilers, like Glasgow Haskell compiler, for implementation of complex processes like time lifting.

10. REFERENCES

Batty, M., M. Dodge, B. Jiang, and A. Smith, 1998. GIS and Urban Design, Center for Advanced Spatial Analyses – CASA. http://www.casa.ucl.ac.uk/urbandesifinal.pdf (accessed Jan, 2005)

Bendikt, M.L., 1979. To take hold of space: isovists and isovist fields, *Environment and Planning B,* (6), pp. 47-65.

Brown, M.G., 2001. Design and Value: Spatial Form and the Economic Failure of a Mall, 3rd International Space Syntax Symposium, Atlanta, USA.

Jiang B., C., Claramunt, and B., Klarqvist, 2000. An Integration of Space Syntax into GIS for Modelling Urban Spaces, *International Journal of Applied Earth Observation and Geoinformation*, (2), pp.161-171.

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

Frank, A.U., 2005. *A Theory for Geographic Information Systems*, Unpublished Manuscript.

Frank, A.U., 2003. *Ontology for spatio-temporal databases'. In Spatiotemporal Databases: The Chorochronos Approach.*

(Koubarakis, M.e.a., ed.), Lecture Notes in Computer Science, Berlin, Springer-Verlag, pp: 9-78.

Frank, A.U., 1998. GIS for Politics, GIS Planet 1998, Lisbon, Portugal, IMERSIV.

Franklin, W.M., 1984. Cartographic Errors Symptomatic of Underlying Algebra problems, International Symposium of on Spatial Data Handling, Zurich, Switzerland.

Herring, J.R., M.J., Egenhofer, and A.U. Frank, 1990. Using category theory to model GIS applications, in Proc. 4th international symposium on spatial data handling, Vol. II, Zurich.

Hillier, B., and Netto V., 2001. Society Seen Through the Prism, 3rd International Space Syntax Symposium, Atlanta, USA.

Hillier, B., 2001. A Theory of the City as Object: How spatial laws mediate the social construction of urban space?, 3rd International Space Syntax Symposium, Atlanta, USA.

Hillier, B., 1996. *Space is the Machine*, Cambridge University Press, Cambridge.

Hillier B., and J. Hanson, 1984. *The Social Logic of Space*, Cambridge University Press, Cambridge.

Lynch, K., 1992. *The Image of the City*, The MIT Press.

Thompson, S., 1998. *The Craft of Functional Programming*, Second Edition, Addison – Wesley Press.