Rezayan, H., A.U. Frank, Farid Karimipour, and M. R. Delavar. "Temporal Topological Relationships of Convex Spaces in Space Syntax Theory." Paper presented at the International Symposium on Spatio-Temporal Modeling '05, Beijing, China, 27.-29.08.2005 2005.

H. Rezayan [a,*], A. U. Frank [b], F. Karimipour [a], 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, Convex Space, and Graph Topology.

ABSTRACT:

GI science development has to be served by an effective GI theory. Development of GI theory requires clear characterisation of GI domain to demonstrate real world effectively and also provide a framework for delineation of rational hypotheses. This requirement is met in recent researches through using mathematics as the basis of GI theory. Axiomatic set theory and formal logic are the foundations of this mathematical approach to study the structures, changes, and spaces in GI domain. While hypotheses require to be modelled into logical structures and be prepared to be evaluated, computer science is also adopted as another foundation of GI theory. One of the recent trends in GI theory development is characterizing the real world as functions and using category theory and algebras as the mathematical basis for handling realities and developing hypotheses. Development of unique and integrated basis for handling static and dynamic GI concepts is one of the hypotheses which are studied in some of these researches. Their outcomes illustrate theoretic feasibility of defining morphisms, known as functors or time liftings, between static and dynamic domains. The time lifting approach was evaluated for some GI models, however, more different models still have to be evaluated. This paper studied time lifting for a topological characterization of convex spaces in real world which is 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. Also some morphologic analyses are defined for deriving the graph's properties which illustrate how the space and time are overcome by the relational systems and convex spaces. Investigation of temporality in Space Syntax theory has shown that more dynamicity exist among activities in local scale. Then the specific problem of this paper is defined as modelling integrated static and dynamic analyses of an activity based scenario in local scale and studying how effective these activities overcome space and time. The derived model is implemented for analysing a simulated urban public transportation system using a functional programming language known as Haskell. The successful implementation validated the time lifting approach for topological models of convex spaces, as the main aim of this paper. Besides, questions are emerged about mixed usage of static and dynamic data and level of computation's time and memory growth rates.

1. INTRODUCTION

Is it right that over 80 percent of information has spatial factors? Successive introduction of spatial rules as important foundations of theories in different sciences strengthens the rightness of this claim. For example in architecture science, Space Syntax theory is emerged 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 whether the spatial laws, derived out in different sciences independently, are consistent and could they be integrated together?

GI 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 spaces. This mathematical trend is generally based on axiomatic set theory and formal logic. This trend is also accompanied with addressing the derived concepts of GI theory in computer science.

Then it would be truly supposed that GI concepts which have definite mathematical and algebraic structures (like topology) could take advantages of GI theory at once, while other GI concepts need to be redefined. 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 following functionalists' approach, especially the results provided by Frank and his colleagues, for developing GI theory by adopting category theory and algebras for time formalization. This formalization is used for evaluation of spatio-temporal concepts integration in Space Syntax theory. These concepts are resulted from investigation of convex spaces' topological relationships in human settlements.

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

* Corresponding author.

The paper is composed of 9 sections. In section 2, Space Syntax theory is reviewed. Topological properties of this theory and their morphologic analyses are described in section 3. Then, temporality in Space Syntax theory is investigated in section 4. In section 5, formalization of time in GI theory is described. Section 6 provides a specific problem definition which is then implemented in section 7 and proceeded to a case study in section 8. Finally 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 directions 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 emphasises on physical attributes that usually restrict its scale of operation to arrangements of streets, buildings, and landscapes (Batty et al., 1998). Architecture, design, and planning are suffered from lack of scientific theories, as existing theories are mostly normative and weakly analytical (Hillier, 1996).

Space Syntax theory is a spatial theory which attempts to overcome the mentioned theoretical deficiencies by providing 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 spaces.

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

One of the central concepts of Space Syntax theory is urban grid. It is the pattern of public space linking the buildings (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 the relational systems of societies are strongly connected space-time relational systems which their individual relations are space-time relations and events / activities (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 required information to reproduce cultural patterns of space is found in the spatial configurations themselves as relations / interactions. Individuals who make up such an 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 movement in societies (Hillier and Netto, 2001).

Then it is depicted that the social construction of space in human settlements is mediated by two kinds of spatial laws: those by which different ways of placing buildings gave rise to different spatial configurations (local and conservative); and those through which different spatial configurations create 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 and small (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 Space Syntax theory, residential and cultural factors, which are variants, dominate local scale and commercial and micro-economic factors, which are invariants, form global scale.

Dealing with invariants, Space Syntax theory introduces a universal pattern which could be extracted from the space-time relational systems in global scale. It is known as deformed wheel pattern (Hillier, 2001). This pattern is firstly used for explanation of movement. Also the effect of variants on a society are analyzed studying the level of deformation occurred in its deformed wheel pattern.

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 here as a transformation of space-time and a transformation of society (Hillier and Netto, 2001). This theory generates topological formal models for space-time relational systems of convex spaces in human settlements. These systems are represented as connectivity graphs and equipped with effective methods for analyzing their morphologic properties which are described in Section 3.

3. TOPOLOGICAL RELATIONSHIP OF CONVEX SPACES IN SPACE SYNTAX THEORY

The mentioned graph representations of societies' relational systems are generated as follow:

1. Spatial decomposition of spatial configuration into elementary units of analysis: bounded spaces, convex spaces and axial lines. These are defined as (Brown, 2001):
   - Bounded spaces (typical enclosable rooms) 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 and usually correspond with flows and globalization of space. They connect all incidences of convex spaces based on their inter-visibility. (Figure 1.b).

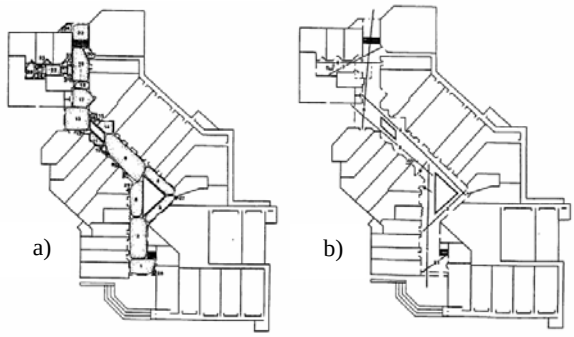

Figure 1. Example of analysis units' extraction in Space Syntax theory for a market (Brown, 2001)

2. Representing derived analysis units and their connections as a connectivity graph in which its nodes and links are respectively the analysis units and their connections. These graphs are usually big, shallow, non-dendritic, highly integrated, and everywhere ringy (with a large number of cycles) rather than tree-like (Hillier and Netto, 2001). Jiang et. al (2000) illustrates three different representations for a connectivity graph depending on the degree of linearity in 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).

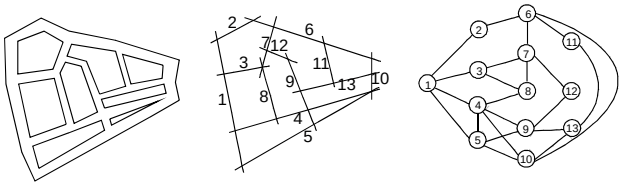

Figure 2. Axial representation (Jiang et al., 2000)

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

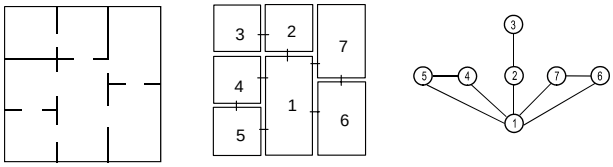

Figure 3. Convex representation (Jiang et al., 2000)

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

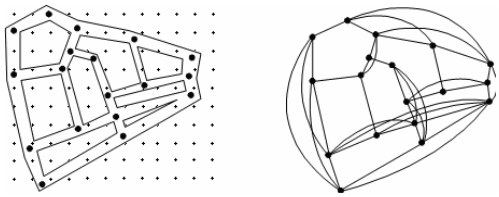

Figure 4. Grid representation (Jiang et al., 2000)

3. Deriving the graphs morphologic properties: connectivity, control value, depth, and integrability. These are defined as follow (Jiang et. al, 2000, Brown, 2001):

– The connectivity value is the number of immediate neighbours of nodes (1).

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

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 (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).

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

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 is 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).

$$RA_i = 2(MDi - 1)/(n\text{-}2) \qquad (4)$$

where RA_i = Relative asymmetry value of ith node

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 graph based on a target node. This process is defined as creating a justified graph (j-graph) for a node. J-graphs are viewpoints of individuals to 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 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 (Hillier and Netto, 2001).

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: global and local. 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 (Section 2). Therefore, movement at global scale is known as general movement which is highly governed by invariant rules like commercial and micro-economic rules (Hillier, 2001).
While that Space Syntax theory deals with space-time as a genotype (Section 2), the main spatio-temporal question in global scale would be how a society is reproduced 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, not dealing with the 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 kinds of environment's representation (Section 3), especially, axial and convex ones.

4.2 Time in Local Scale

In local scale, we face with discrete and 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, what we need to compare is not so much the contents of these 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 these activities are (Hiller and Netto, 2001):

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 for 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 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 inherent, fundamental and different dimensions of reality in which people live (Frank, 2005; Egenhofer and Mark, 1995).

Despite many efforts and researches carried out for handling space and time, these two did not advance co-ordinately. Effective handling of time is still a controversial and preliminary topic in GI science and technology. Besides lack of effective integration approaches, GI scientists enumerate temporal deficiencies as one of the main troublesome issues of GI science and technology (Frank, 2005). Time handling problems could be detailed as follow:

- Lack of a comprehensive and basic spatio-temporal ontology (Frank, 2003).
- Dealing with time as a discrete or partial continuous property of world while our unique reality (space and time) is continues and governed by differentiable laws (Frank, 2003).
- Dominance of analytical approaches, suffered from 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 (Herring et al., 1990).

A general reason for these deficiencies is lack of effective GI theory. GI theory, like theories in 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 from introduction of GI science (Goodchild, 1992) and dealing with GI theory development, recently, it is hypothesized that mathematics can provide the required basis of GI theory. Mathematics is treated in GI theory as study of structures, changes, and spaces (further than figures and numbers) and used following 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 to 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 functional approaches for definition of GI related interactions is provided by Frank (2003) as the closed loop of semantics. He described interactions as composition of a series of functions carry out by agents in real world (Figure 4). These functions could be expressed as (5):

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

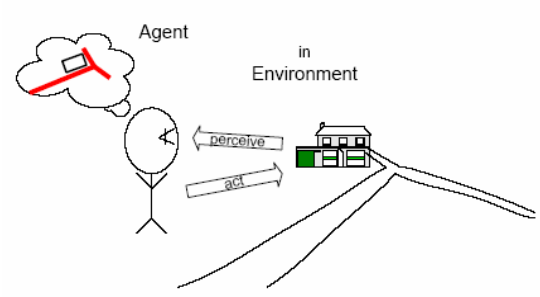

Figure 4. Closed loop of semantics

The commutative version of the closed loop of semantics is as follow (6):

$$\begin{array}{ccccc} Reality & \rightarrow & g & \rightarrow & Reality \\ \downarrow & & & & \uparrow \\ Observation & \rightarrow & g' & \rightarrow & Act \end{array} \qquad (6)$$

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 GIS by Herring et al. (1989).

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

Category theory is introduced by Herring et al. (1989) as the mathematical basis which could support the functional representation of GI interactions. This is a mathematical theory that abstractly (free from 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 (*obj(C)*), operations are morphisms 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 *field category* consists of *fields* as objects and *field homomorphism* as its morphisms (8).

$$\begin{aligned} & m : G \rightarrow H \\ & m(u + v) = m(u) \mathbin{++} m(v) \\ & m(u * v) = m(u) \mathbin{**} m(v) \end{aligned} \qquad (8)$$

where G and H = Fields like *(G, +, *)* and *(H, ++, **)*
m = Homomorphism function
u and v = Members of G

A functor is a morphism which associates elements and operations from one category to another and preserves the operator algebra (Herring et al., 1989). Functor F from category C to category D 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. This means that category theory can be described in itself. While many mathematical theories attempt to study a particular type of structure just by relating it 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 in moving from static to dynamic domain results in more complixity.
By defining all GI concepts as formal mathematical structures and 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, using morphisms within and among categories.

5.2 Functional Formalization of Time

Considering C_s and C_d as two categories which carry one kind of mathematical structure in their static and dynamic domains respectively, a functor from the static category to the dynamic one can lift us to temporal domain. This functor will take a function that can be used for static data and generate a function could be used for dynamic data. This kind of transformation from static to dynamic domain is usually mentioned as *time lifting*. While it is possible to define a unique time lift function, for simplicity, different time lift functions are illustrated here (9), using lambda calculus syntax, for functions with 0 to 2 parameters.

$$\begin{aligned} & > lift0\ a = \backslash t \rightarrow a \\ & > lift1\ op\ a = \backslash t \rightarrow op\ (a\ t) \\ & > lift2\ op\ a\ b = \backslash t \rightarrow op\ (a\ t)\ (b\ t) \end{aligned} \qquad (9)$$

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

For example time lifting of a two parameter function could be done using the *lift2* function (10).

$$\begin{aligned} & f : a \rightarrow b \rightarrow c \\ & \\ & g : (t \rightarrow a) \rightarrow (t \rightarrow b) \rightarrow (t \rightarrow c) \\ & g = lift2\ f \end{aligned} \qquad (10)$$

where a, b, and c = Static types
(t→a) and *(t→b)* = Dynamic input functions
(t→c) = Dynamic output functions
example: 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 values could be structured as a dynamic one being defined as a function from time to the value (11)

$$> Changing\ v = Time \rightarrow v \qquad (11)$$

where v = Any type
Time = Time parameter

These dynamic types could be used in functions, like their static counterparts, based on polymorphism. However, this depends upon feasibility of instantiating the dynamic types for classes used by static types. This process is known as overloading or ad-hoc polymorphism. After overloading of dynamic types, 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 computers' float numbering systems 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 extensionally by validating definition of infinite series of numbers. Lazy evaluation means extensional computation of statements and just evaluating required parts of the statements (12).

```
> naturals = [1, 2 ..]                    (12)
> n_5 = take 5 naturals = [1,2,3,4,5]
```

where *naturals* = Infinite series of natural numbers

The general solution for round off error problem, which is proposed by Franklin (1984), is utilization of rational numbering system. 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*** 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 numbering 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. While overloading is just feasible in computing environments conforms to object oriented architecture, the existing computing environments are either imperative or do not support overloading for structured data types, especially for 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 (13).

$$> Changing\ (Ration\ Integer) = Time \rightarrow (Ratio\ Integer) \qquad (13)$$

In continue the following synonyms (14) will be used:

```
> type RI = Ratio Integer                 (14)
> type CRI = Changing (Ratio Integer)
```

The static and dynamic rational types are overloaded over a *Field* class contains basic numeric operators (15).

```
> class Field a where                     (15)
>         (+),(-),(*),(/) : a → a → a
```

Overloaded instances for static and dynamic types are:

```
> instance Field (RI) where               (16)
>         ...

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

Then any functions, which use these basic operators, would be valid for both static and dynamic types (17).

```
> dm : (Field a) ==> a → a → a            (17)
> dm a b = (a + b) * (a – b)
```

example:

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

6. PROBLEM DEFINITION

Considering the previously mentioned issues, the aim of this paper can be restated as "implementing integrated analyses for static and dynamic topological relationships of convex spaces in Space Syntax theory using the time lifting approach". Dealing with this, the grid representation is selected as the basis of analyses (Section 3). In grid representation rather simpler approach for deriving convex spaces is provided than the two other representations, as the vantage points are predefined and finite (set of regular grid points). These points can be proposed as potential incidences of convex spaces and their linkages can be derived by checking their intervisibilities.
Moreover, the grid representation supports more rapid dynamicity, movement, and changes of activities in local scale (Section 4.2). Then our problem is limited to intervisibility analysis in a definite and finite point set which represents positions of static and dynamic activities. The intervisibility analysis would be carried out within an underlying static environment. This static environment consists of passages (e.g. streets) and barriers (e.g. buildings) which enable or disable movement and intervisibility.
The resulted connectivity graph will be composed of positions of static and dynamic activities as nodes and their intervisibility relations as links.

It is expected that this approach could support analyses of different kinds of dynamic activities, especially public services in cities (e.g. transportation, safety, and security), in regards with evaluating how effective they interact, overcome and cover space and time.

7. IMPLEMENTATION

The required components for implementation are defined as follow:

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

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

7.1 Dynamic Points

A general point is defined as an algebraic data type (18).

```
> data Point a = Point Id a a                                    (18)
```

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 with rational coordinates. A dynamic point type with dynamic rational coordinates would be defined as *Point (CRI)*, too.
Considering that multiplication and division of points are not required, the basic functions for point data type are defined as a class denoted as *Points* (19).

```
> class Points p s where                                         (19)
>         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 default definitions, just *(+)* and *(-)* functions are overloaded for static (20) and dynamic (21) point data types.

```
> instance Points Point a where                                  (20)
>         (+) (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                       (21)
>         (+) = lift2 (+)
>         (-) = lift2 (-)
```

where _ = Any value

The other basic operations such as equality and ordering of points could be defined similarly (omitted here).

7.2 Connectivity Graph

The proposed connectivity graph is a set of binary relations between points (22) (Thompson, 1998).

```
> data Set a = Set [a]                                           (22)
> type Relation a = (a,a)
> 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

Nodes are also defined as an algebraic data type (23).

```
> data Node = Node Id ([Id], C, Ctrl, D, MD, RA)                 (23)
```

where *Node* =Type constructor of a node
Id = Unique identifier for a node which is the same as id of its corresponding point.
[Id] = List of connected nodes' ids
C, Ctrl, D, MD, and *RA* = Static parameters as connectivity (C), control (Ctrl), total depth (D), mean depth (MD), and integration (RA).

Nodes manipulation functions are defined as some *set* and *get* functions (24).

```
> setNodeId :: Node → RI → Node                                  (24)
> setNodeId (Node id param) id' = Node id' param

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

> getConnections :: Node → [RI]

> setConnections :: Node → [RI] → Node

> getConnectivity, getControl, getTotalDepth,
> getMeanDepth, getIntegrability :: Node → RI

> setConnectivity, setControl, setTotalDepth,
> setMeanDepth, setIntegrability :: Node → RI → Node
```

In following sections a graph with static numbers is used which is denoted as *Graph RI*.

7.3 Static Environment

As mentioned in section 6, the required static environment defines movement barriers and passages for dynamic activities. This could be modelled as a set of planar polygons which do not intersect or include each other (25). Spaces between these polygons define the passages.

```
> data Environment a = [Polygon a]                    (25)
```

where *Environment* = Type constructor of an environment

A polygon is defined by its bounding straight line segments named as *Lines*. A *Line* (26) and a *polygon* (27) are also defined as algebraic data types, too.

```
> data Line a = Line (Point a) (Point a)              (26)
```

where *Line* = Type constructor for a line

```
> data Polygon a = Polygon [Line a]                   (27)
```

where *Polygon* =Type constructor for a polygon

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

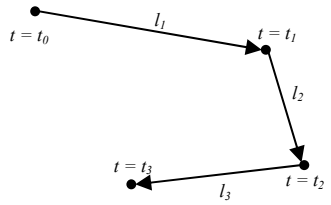

Figure 6. A path consists of three directional straight lines at different times

Then a path could be constructed as a time based conditional statement of multiple directional straight lines (28).

```
> path = cond [c1, c2, ... ] [l1, l2, l3, ...]        (28)
```

where *cond* = Checks a series of conditions finding the first valid condition and selecting its respective value

$c_1 = t_0<t<t_1$
$c_2 = t_1<t<t_2$
l_1, l_2 and l_3= Line equations

The proposed movement of dynamic activities would be defined by these paths.

7.4 Functions

Most of the defined functions are separated in two parts:

1. functions for component analysis
2. functions for generalization of the component analysis functions for lists analysis, which are defined by prefix *m*

areIntervisible function is one of the basic functions which checks for existence of intervisibility relation between two points in an environment (29).

```
> areIntervisible :: (Bools b, Points Point a) =>                (29)
                     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 lines intersection counting process, further definitions are omitted here. *deriveIntervisibles* (30) and *deriveRelations* (31) functions are defined for derivation of all intervisibility relations of a point. Also *mDeriveRelations* (32) function is defined for generalizing *deriveRelations* function over all points of a list.

```
> deriveIntervisibles:: (Points Point a) =>                      (30)
      Environment a → [Point a] → Point a → [Point a]
> deriveIntervisibles env ps p = filter (areIntervisible env p) ps
```

```
> deriveRelations:: (Points Point a) =>                          (31)
      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) =>                         (32)
      Environment a → [Point a] → [[Relation RI]]
> mDeriveRelations env ps = map (deriveRelations env ps) ps
```

Other functions are also used for flattening the result of *mDeriveRelations* function into one list of relations and also for removing duplicate relations. These functions are omitted here. Then, *cGraph* function (33) is defined to construct the connectivity graph recursively.

```
> cGraph :: (Points Point a) =>                                  (33)
          Environment a → [Point a] → Graph RI
> cGraph env ps = Set (mDeriveRelations env ps)
```

Graph nodes are generated from the points list using *makeNodes* function (34).

```
> makeNodes :: (Points Point a) => [Point a] → [Node]            (34)
> makeNodes [] = []
> makeNodes (p:ps) =
                (Node (pID p) ([],0,0,0,0,0)) : makeNodes ps
```

The morphologic analyses of the graph are defined as follow:

1. Connections list (35 and 36):

```
> deriveConnections:: Graph RI → Node → Node                    (35)
> deriveConnections g n = [b | (a,b) ← g, a == (getNodeID n)]
                        ++ [a | (a,b) ← g, b == (getNodeID n)]

> mDeriveConnections:: Graph RI → [Node] → [Node]               (36)
> mDeriveConnections g ns = map (deriveConnections g) ns
```

2. Connectivity value (37 and 38):

```
> connectivity :: Node → Node                                   (37)
> connectivity = setConnectivity.length.getConnections
```

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

```
> mConnectivity:: [Node]→ [Node]                                (38)
> mConnectivity ns = map connectivity ns
```

3. Control value (39, 40, and 41):

```
> control :: Graph RI → Node → Node                             (39)
> control g n = setControl (sum (map reci.getConnectivity
(findNodes g (getConnections n))) n
```

where *findNodes* = Gets ids and returns their relevant nodes.
reci = Returns a reciprocal number.
sum = Returns summation of a list of numbers.

```
> mControl :: Graph RI → [Node] → [Node]                        (40)
> mControl g ns = map (control g) ns
```

4. Depth value (41, 42, 43, 44, and 45):

```
> depth :: Graph RI → Node → Node → RI                          (41)
```

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

```
> totalDepth :: Graph RI→ Node → RI                             (42)
> totalDepth g p =
    setTotalDepth (sum [depth g p q | q ← graph]) p

> mTotalDepth:: Graph RI → [Node] → [Node]                      (43)
> mTotalDepth g ns = map (totalDepth g) ns

> meanDepth :: Graph RI → Node → Node                           (44)
> meanDepth g n =
    setMeanDepth ((getTotalDepth n) / (length g – 1))

> mMeanDepth :: Graph RI → [Node] → [Node]                      (45)
> mMeanDepth g ns = map (meanDepth g) ns
```

5. Inegrability value (46 and 47):

```
> integrability :: Graph RI → Node → Node                       (46)
> integrability g n=
    setIntegrability (2 * (getMeanDepth n - 1) / (length g – 2))

> mIntegrability:: Graph RI → [Node] → [Node]                   (47)
> mIntegrability g ns = map (integrability g) ns
```

Then all the morphologic analyses can be composed (48).

```
> calcParam :: Graph RI → [Node] → [Node]                       (48)
> calcParam g = ((mIntegrability g).
                 (mMeanDepth g).(mTotalDepth g).
                 (mControl g).mConnectivity).
                 (mDeriveConnections g)
```

Finally, all functions are composed into a function denoted as *analyseGrid* (49).

```
> analyseGrid :: (Points Point a) =>                            (49)
        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 10 will generate and analyse graphs at 10

8. CASE STUDY

The case study is defined as analysing the quality of overcoming space and time by a simulated public bus transportation system in a city. This scenario is implemented in a Haskell compiler known as Hugs.
The city environment is implemented with five static polygons. Also seven buses are implemented as dynamic activities (Figure 7). Two sample definitions of these buses are presented in (50).

```
> Activity1 = cond [] [l1_s1]                                   (50)
> 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 one straight line, path of *Activity2* has two connected straight lines which are controlled by one temporal condition. These two paths are shown in Figure 7 as dashed arrows.

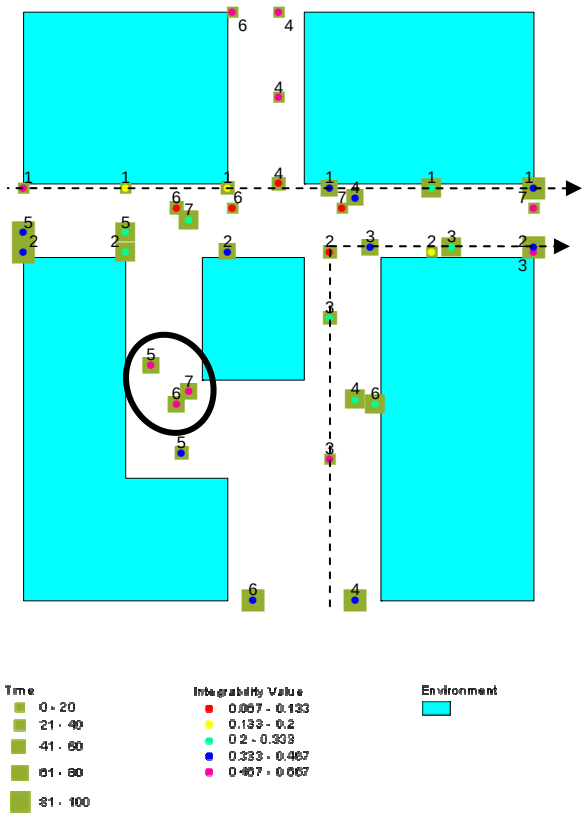

Figure 7. Simulated and analysed bus transportation system for the case study

The mentioned *analyseGraph* function (49) is used in this case study at 0, 25, 50, 75, and 100. The results are presented in Figure 7 as graduated box symbols. Also the calculated integrability value is selected arbitrarily to be shown over time.
As shown, the result could be used for defining areas with different level of overcoming space and time and then modifying the paths and time schedules to obtain more effective coverage. Other spatial and socio-economic analyses are also possible based on the concepts provided by Space Syntax theory. For example, a region of high integrability between times 50 and 70 is shown in Figure 7 by a thick ellipse. This area could absorb high level of travelling demand during the mentioned duration and this interpretation could be the basis for further treatments.

9. CONCLUSIONS

The implementation carried out in this paper determines validity of the functional approach for time lifting the topological relationships of convex spaces. The proposed topological characterization is firstly divided into two processes: one for deriving analysis units and generate the topological structure and the other for properties extraction. While the former process is time lifted the latter is remained unchanged. Then they are composed as a unique time lifted process. This approach could be adopted for time lifting of other topological structures.
Besides, Space Syntax theory conformance with GI concepts and its consistency to be handled and integrated with GI theory and likely other spatial theories is shown. Also, the mentioned specific approach in modelling dynamic activities in local scale and analysing how effective they overcome space and time could be widely used, especially for urban related public services.
The succinct, comprehensible, and testable functions defined in our implementation represent suitability of functional programming languages more for evaluating GI theory hypotheses.
Some considerations are also required about mixed usage of static and dynamic data and clarification of computation time and memory growth rates during time lifting. While functional programming environments try to manage memory automatically, more specific techniques for controlling memory allocations and garbage collections are needed. All these imply utilization of more sophisticated techniques and compilers, like Glasgow Haskell compiler.

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)

Benedikt, 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.

Goodchild, M. F., 1992. Geographical information science. *International Journal of Geographical Information Systems*, 6(1), pp. 31–45.

Grenon, P., and B. Smith, 2004. SNAP and SPAN: towards dynamic spatial ontology. *Spatial Cognition and Computation*, 4(1).

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.