Frank, A.U. "Spatial Communication with Maps: Defining the Correctness of Maps Using a Multi-Agent Simulation." In *Spatial Cognition Ii (International Workshop on Maps and Diagrammatical Representations of the Environment, Hamburg, August 1999)*, edited by Ch. Freksa, W. Brauer, Ch. Habel and K. F. Wender, 80-99. Berlin Heidelberg: Springer-Verlag, 2000.

Spatial Communication with Maps: Defining the Correctness of Maps Using a Multi-Agent Simulation

Andrew U. Frank

Dept. of Geoinformation
Technical University Vienna
frank@geoinfo.tuwien.ac.at

Abstract. Maps are very efficient to communicate spatial situations. A theoretical framework for a formal discussion of map production and map use is constructed using a multi-agent framework. Multi-agent systems are computerized models that simulate persons as autonomous agents in a simulated environment, with their simulated interaction. A model of the process of map production and map use is constructed based on a two-tiered *reality and beliefs model*, in which facts describing the simulated environment and the simulated agents' beliefs of this environment are separated. This permits to model errors in the persons' perception of reality.

A computerized model was coded, including all operations: the observation of reality by a person, the production of the map, the interpretation of the map by another person and his use of the knowledge acquired from the map for navigation, are simulated as operations of agents in a simulated environment.

1 Introduction

Daily experience tells us that maps are a very efficient and natural way to communicate spatial situations. Small children produce maps spontaneously and maps are among the earliest human artifacts. However, we seem not to have a good understanding how maps communicate spatial situations. Formal models for the processes of map production and use are missing. This leaves judgment of map quality to a large degree subjective, as map construction and map reading are both implying intelligent human interpretation. A person using a map knows the general morphology of the terrain and uses this knowledge to draw appropriate conclusions from the graphical signs on the map. Unfortunately, this general assumption of intelligent interpretation breaks down in unfamiliar terrain when a map is most needed. A more objective measure for correctness of a map, which does not rely on additional knowledge, is required. So far, we can only define consistency of a database as the absence of internal contradiction in a data quality. A formal definition for correctness, i.e., the correspondence between

data and reality, cannot be constructed, as it would need to bridge between reality and the formal representation.

Formal methods to define correctness of geographic data are urgently needed in the emerging business with geographic information. It is necessary to assess the quality of geographic data collections and compare them. For example, we must be capable of comparing the quality of competing data providers for In-Car Navigation Systems and point out errors based on objective criteria and not just based on anecdotes. The unfortunate adventure of a car driver following the advice of his In-Car navigation system to cross a river over a bridge was widely published. Too late, when the car was already floating in the river, he noticed that there was no bridge but only a ferry!

In this paper a multi-agent formalism is used to produce a model of map production, map communication and map use. Multi-agent systems are computerized formal models, which contain a modeled environment and autonomous agents, which interact with this environment (Ferber 1998; Weiss 1999). The model formalizes the processes involved (Figure 1); it is not intended to be used for actual navigation in a city. I use here a simple task to make the discussion concrete; namely, the production and use of a small street network map for navigation. The model constructed simulates:

- The environment, which is constructed after the example of a small part of down-town Santa Barbara (Figure 4);
- A map-maker who explores the environment and collects information, which he uses to construct a map of the area; and
- A map-user who acquires this map to gain knowledge, which he uses to navigate in this environment.

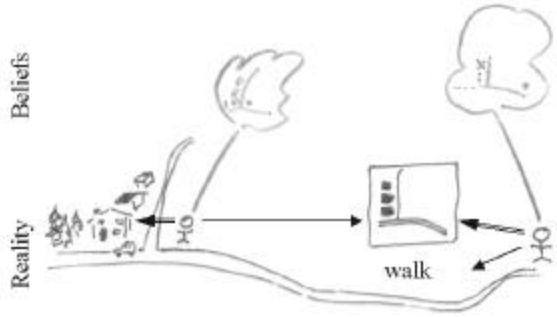

Fig.1. An agent producing a map and another agent using a map for navigation

The environment represents the world in which persons live and the agents represent the persons who make and use maps (Figure 2). The simulation includes multiple agents – at least one map-making agent and one or several map-using agents. In the simulated environment, it is possible to define what it means that a map is correct and how to compare the effectiveness of map communication with verbal communication. The definitions point out the strong connection between correctness of a map and the intended use of a map or spatial data set.

Multi-agent systems have been used previously for map generalization (Baeijs, Demazeau et al. 1995), where different agents apply rules to a given map graphics to produce a generalized map. The approach here is very different; we do not intend to model a part of the map production process, but to model the complete process, which starts with data collection in reality, produces the map and then the process of map use: reading the map to navigate in an unknown territory.

Real World Situation	**`Multi-Agent Model`**
Real World Situation	`Model`
World	`Environment`
Person	`Agent`
Map-maker	`Map-making agent`
Map user	`Map using agent`
Fact	`Belief`

Fig.2. Mapping from reality to model

Braitenberg has introduced computational models in psychology (Braitenberg 1984) and demonstrated how insight can be gained from a fully simulated (synthetic) model. The agents used here are nearly as simple as Braitenberg's "vehicles"; they are sufficient to contribute to our understanding of correctness and effectiveness of maps. *Correctness* of the map is judged within the model as the success of the agents in navigating in the environment. *Effectiveness* of the map can be judged by comparing the size of different representations to communicate the same information between agents. One can demonstrate that verbal descriptions are equally effective if one has to communicate a single route, but are inefficient to communicate a complex spatial situation, e.g., the street segments in a downtown area.

To construct a computational model for map production and map use is novel. The model uses a two-tiered *reality and beliefs representation*, in which reality (facts) and the agents' cognition (beliefs) are represented separately (Figure 1). Errors in the agent's perception of reality or errors in the production or reading of artifacts like maps, representing and communicating an agent's (possibly erroneous) beliefs can be modeled. It is possible to include imaginary or contrafactual maps as well. The formalization is in an executable functional language, using Haskell (Hudak, Peyton Jones et al. 1992; Peterson, Hammond et al. 1997; Peyton Jones, Hughes et al. 1999). The code is available from the web (http://www.geoinfo.tuwien.ac.at) and extensions to investigate similar questions are possible.

2 Computational Models Separating Reality and Beliefs

Computational models are formalized methods to describe our understanding of complex processes. They have been extremely successful in many areas, especially modeling aspects of our physical environment where models of reality and models of processes are linked (Burrough and Heuvelink 1988). They have been less successful in the information domain, in my opinion because the linkage between the static data describing reality and the processes of data observation, data collection and data use has not been achieved (Frank 1998). The described two-tiered *reality and beliefs*

computational model, where the simulation contains separate representations of the environment and representations of the agent and its knowledge about the environment as two separate data sets, overcomes this problem, because the observation and data use processes are explicitly included. Following an AI tradition, the agent's knowledge is called 'belief' to stress the potential for differences between reality and the agent's possibly erroneous beliefs about reality (Davis 1990).

The model takes into account many of the often-voiced critiques against formal models of cognition. In terms of Warfield and Stich (Stich and Warfield 1994, p. 5ff) models of cognition must have three properties:

- Naturalness: The semantics of the mental representations are linked to the operations of the agent observing the environment and acting in it. These observation operations are part of the model and their properties described.
- Misrepresentation is possible, as the model contains separate representations for the data, which stand for reality, and the data, which represent an agent's beliefs. The models of observation processes may produce errors; the actions may not use the information the agent has correctly represented.
- Fine-grained meanings are achieved, as concepts and what they are linked to in reality are separate. It is possible that the agent maintains beliefs about two different concepts, only later to find out that the two are the same.
- The model constructed here has these properties:

2.1 Naturalness and Semantics

The semantics of the mental operations on the beliefs are directly connected to the person's bodily actions (Johnson 1987): mentally following a street segment's mental representation is given meaning through the correspondence with the physical locomotion of the agent along a street segment. This correspondence is kept in the model; the simulated mental operations of the agents are linked to the simulated bodily actions of the agents. The model is therefore not disembodied AI (Dreyfuss 1998) because the linkage between bodily actions of the agents and their mental representation is direct and the same as in persons (Lakoff and Johnson 1999).

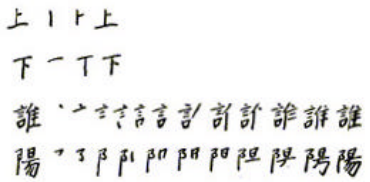

Fig. 3. Instructions how to draw Chinese characters (two simple and two complex ones from Tchen 1967)

"Information itself is nothing special; it is found wherever causes leave effects" (Pinker 1997, p. 65-66). The map product can be seen as the sequence of drawing steps (the causes) the map-maker follows to produce it and the map-reader does retrace these steps in his map reading process. It is instructive to observe that Chinese

characters are not learned as figures but as a sequence of strokes (Figure 3). This is not only important for production, but also for recognition of signs created at different levels of fluidity. Westerners often copy Chinese characters as a picture and produce images, which are difficult to recognize.

In the multi-agent model, the structure of the operations for locomotion along a street segment, for drawing a street segment or for following a drawn street segment and for mentally following the belief about a street segment can be coded as the same polymorphic operation, applicable to different data structures; e.g., maps, real streets, etc. (not stressed in this presentation).

2.2 Misrepresentation

Persons – both the map-maker and the map-user – can make errors in the perception and form erroneous beliefs about the environment. The maps produced can also have errors or the map reading operation can include errors into the beliefs map-users form about the environment. Such errors or imprecisions can be modeled in the beliefs of the agents. Eventually, agents are prohibited to achieve 'impossible' states of the environment and are stopped in the model from executing impossible actions; e.g., to travel along a street not present in the environment.

2.3 Fine-Grained Meaning

Concepts can have various levels of detail – they can be 'read' from a map and therefore have no experience, e.g., a visual memory associated, or can have a partial knowledge, e.g., a street segment can have a known start but a not yet known end. It is possible to realize later that two different concepts are linked to the same real object, e.g., the intersection where 'Borders' is and the intersection of 'State Street' and 'Canon Perdido Street', which is the same in Santa Barbara (Figure 4). This is possible in multi-agent models, but not included in the simple model presented here.

3 Focused Discussion Based on an Example Case

The investigation is focused with a specific set of tasks in a concrete environment, namely finding a path between named intersections in a city street network. Research in cartography usually concentrates on transformations applied to maps – mostly discussions of map generalization (Weibel 1995) – situated in a diffuse set of implied assumptions about the intended map use and the environment represented (Lechthaler 1999). Concentration on a very specific example avoids this problem. I select here the communication with maps about a city environment for the purpose of navigation,. The environment and the tasks are fixed. Then the construction of a computational model becomes possible.

The *environment* is maximally simplified to make this paper self-contained. It includes street segments, which are connected at street intersections (Figure 4). Many interesting aspects of real cities are left out: for example, one-way streets are excluded as well as turn-restrictions at the intersections; cost of travel is proportional to distance; agents at a node can recognize for all street segments which node they are

connected to, etc. The model of the environment is static, as no changes in the environment are assumed; only the position and beliefs of the agent change in the model. Nevertheless, the model retains the important aspect of exploring an environment and navigatin g in it using the knowledge collected by others. Even from this generalized model, interesting conclusions can be drawn.

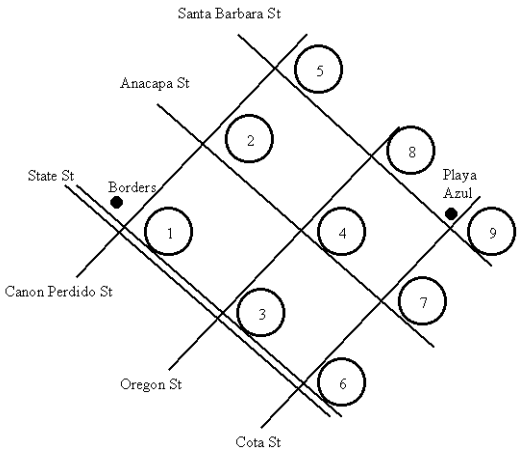

Fig. 4. A small subset of streets of downtown Santa Barbara (with node Ids as used in the code)

Agents are located in this environment at a street intersection oriented to move to a neighboring intersection. They can turn at an intersection to head a desired street segment and can move forward to the end of the street segment they are heading. Agents recognize intersections and street segments connecting them by labels without error. This follows roughly a simplification of the well-known TOURS model (Kuipers and Levitt 1978; Kuipers and Levitt 1990). Operations of the agents simulate the corresponding operations of pers ons in the real world. For example, "observe" applied to agents always means the simulated execution in the model.

The *map-making agent* is exploring the modeled reality and constructs knowledge of each segment traveled and accumulates this knowledge in its memory. From this knowledge, a map is produced as a collection of lines and labels, placed on paper. This map, which looks much like Figure 4 as well, is then given to the agent that represents the map user.

The task *the map-using agent* is carrying out is to navigate between two named street intersections. The agent is constructing knowledge from the map drawn by the map-making agent and then plans the shortest path to the destination using the knowledge gained from the map.

4 Multi-Agent Theory

Multi-agent systems are a unifying theory for a number of developments in computer science. They have interesting applications in robotics, e-commerce, etc., but they also provide a fruitful model to discuss questions of communication and interaction, for example, in Artificial Life research (Epstein and Axtell 1996). Multi-agent systems consist of a simulated environment with which one or more simulated actors interact (Figure 5).

Actors perceive through sensors the environment and have effects on the environment through their actors. The agents are part of the environment and are situated in it. Most authors include direct communication among agents (Ferber 1998; Weiss 1999), but the approach used here models communication between agents as the exchange of artifacts (i.e., a map). The operations of the actors can be described as algebras.

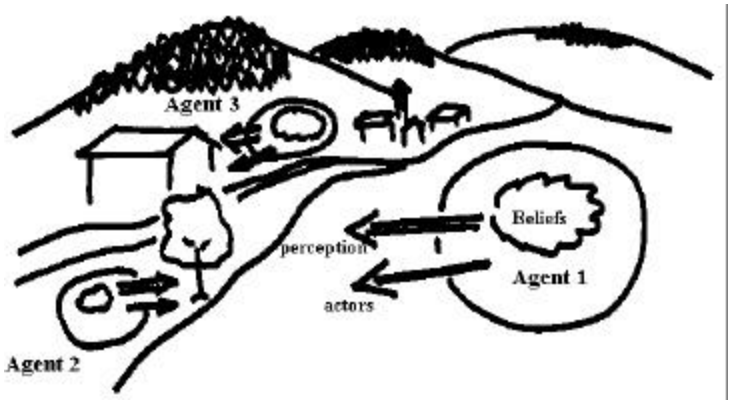

Fig.5. Environment and actors

The representation of this computational model is two-tiered: it separates completely the facts which stand for the environment, i.e., modeled *reality*, and the set of *beliefs* the agents hold about the environment (Figure 6). Maps are produced by the agents and exist in the environment. They can be used ('read') by the agents; they encode the knowledge the map-making agent has constructed during its exploration of the environment and communicates it to other agents.

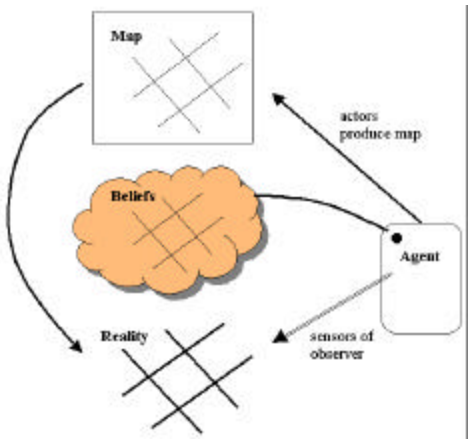

Fig.6. The different kinds of representations

The environment, which represents the world in the model, is encoded by a data structure, which represents the street graph and the locations of the intersections with coordinates. This representation could be extended to include labels for street names and the address range for each side (following the DIME model, which is a widely used representation (Corbett 1975; Corbett 1979)). This constitutes in the model 'reality' – it is therefore by definition complete and correct (and assumed here as static).

Simulated agents observe this environment and form a set of *beliefs* about it. The agents' beliefs may be incomplete, imprecise or even wrong; they are the results of the specific observation process, which is part of the computational model. The agents do usually not have knowledge of the coordinates of locations. For simplicity, only incompleteness of knowledge will be considered here, but investigating the effects of imprecise or vague spatial knowledge (Burrough and Frank 1995; Burrough and Frank 1996) and comparing strategies to compensate for missing information is possible. The agent can produce artifacts, which represent their knowledge. They simulate *Maps* in this environment.

5 Correctness of Maps

Formalizing environment, agents and maps allows us formally define correctness of a map following Tarski as correspondence between the map and the environment, which the map should represent. A representation of reality is correct, when operations in reality have results, which correspond to the results of corresponding operations in the representation. Applied to navigation: a map is correct, if the person using it plans the same path as a person, who knows the environment well, would walk.

This is best expressed as a homomorphism diagram (Figure 7) as usually drawn in category theory (Pierce 1993). It is based on a mapping *f* between two domains (the domain of agent's representation and the environment) and two corresponding opera-

tions *sp* and *sp'* (one to plan a shortest path in the agents' mind and the other walking the shortest path in the environment), such that first mapping the input from one to the other domain and then applying the operation, or first applying the operation and then mapping the result is the same:

$$f(sp(l)) = sp'(f(l)).$$

In general, formalization of Tarski semantics is not possible – the chasm between the world and the representation cannot be bridged. In the multi-agent model, however, both the environment and the agents' beliefs are part of the model and represented (but we gave up the restriction that the environment in the model is representing exactly some real-world situation). The correspondence between an agent's beliefs and the environment is definable as a homomorphism (Figure 7), where objects and operations are set into correspondence: the (simulated) mental act of an agent determining a path and the actual (simulated) walking of the path must be homomorphic. This means that the agent's beliefs about distances must be precise enough to determine the correct shortest path.

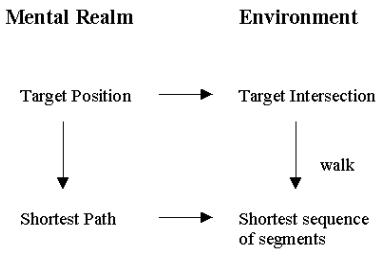

Fig. 7. Homomorphism between Real and Mental Representation

The processes of map making and map use are combinations of homomorphisms (Figure 8). The construction of a map is based on the (correct) mental representation of the map-maker gained through exploration of the environment. The mental representation of the map user is then constructed while reading the map. If each square in Figure 8 is a homomorphism, then a homomorphism from begin to end applies (Walters 1991). In all cases, the homomorphism maps not only between the simulated objects of the environment (street segments, intersections, respectively lines and points on the map), but also between the corresponding simulated operations of the agents (walking a street segment, imagine walking the street segment, drawing a line, etc.).

For complex decisions it may be difficult to show that these mappings are homomorphic. Inspecting the algorithm used to determine the shortest path (Dijkstra 1959; Kirschenhofer 1995), one finds that only two operations access the representation of the data. Initially, all the nodes in the graph must be found (other forms of the algorithm need the nodes connected to a given node) and the distance between two nodes. It is, therefore, only necessary to show that these operations are mapped correctly between the domains (see section 8).

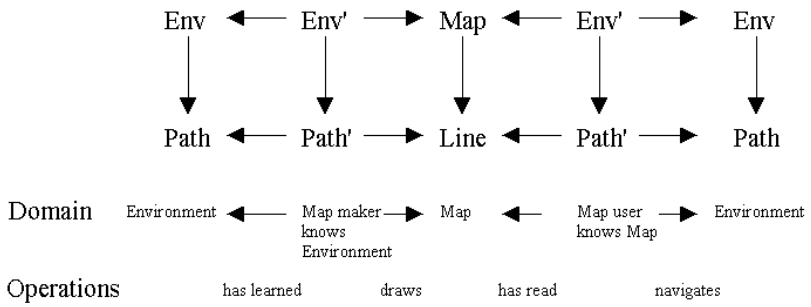

Fig.8. Combinations of homomorphisms

6 Formalization

For a multi-agent model, we have to represent the environment and the agents together with their interactions. We select here an algebraic approach and define classes with operations. To implement the model, data representations are also given. The formalization uses the Haskell language syntax (Peyton Jones, Hughes et al. 1999) to describe algebras (as classes) with the operations defined for the parameterized types and representations (as data or type).

The next subsections show the abstractions selected for street environment, the agents, map-makers, maps and map-users. For each type of object, the necessary operations are described. The representations selected here are given for illustration purposes and the goal is simplicity of the presentation. They document what information must be available, but any other representation for which the algebras given can be implemented would serve as well. No claim is made that the representations given here resemble the representations used in human mental operations

6.1 Environment

The simulation is in an environment, which contains the agents with their beliefs, the street network, and, for simplicity, a single map. It can be represented as a data structure, consisting of

```
data Env = Env [Intersection] [Agent] Map
```

The next subsection define now the data for these three parts:

6.2 Static Street Environment

The street network consists of s treet segments (*edges*), which run from an intersection to the next. The intersections are called *nodes* and the street network is represented as a graph. The algebra for the street-network must contain operations to determine the

position of a node as a coordinate pair (*Vec2* data type), test if two nodes are connected and find all nodes, which can be reached from a given node (operations *connectedNodes*); the shortest path algorithm requires to find all nodes and to get the distance between two nodes. Two operations to add a node and to add a connection to the network are also included.

```
class Streets node env where
            position :: node -> env -> Vec2
            connected :: node -> node -> env -> Bool
            travelDistance :: node -> node -> env -> Float
            connectedNodes :: node -> env -> [node
            allNodes :: env -> [node]
            addNode :: (node, Vec2) -> env -> env
            addConnect :: (node, node) -> env -> env
```

Nodes are just numbered (Figure 4) and Intersections consist of the Node (the node number as an ID), the position (as a coordinate pair) and a list of the connected node numbers.

```
data Intersection = IS Node Vec2 [Node]
data Position = Position Node Vec2
data Node = Node Int | NoNode
data Vec2 = V2 Float Float
```

6.3 Agents

The agents have a position at a node and a destination node they head to. They can either move in the direction they head or can turn to head towards another destination. They are modeled after Papert's Turtle geometry (Papert and Sculley 1980; Abelson and Disessa 1986)). After a move, the agent heads to the node it came from (Figure 9). This behavior can be defined with only four axioms:

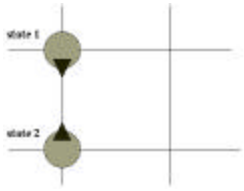

Fig. 9. The position of an agent before (state1) and after a move (state2)

1. Turning (*changeDestination*) does not affect the position:
   pos (a, (changeDestination (a,n,e)) = pos (a,e)
2. Moving brings agent to the node that was its destination:
   pos (a, move (a,e)) = destination (a,e)
3. The destination after a move is the location the agent was at before the move:
   destination (a, move (a,e)) = pos (a,e)

4. Turning (*changeDestination*) makes the agent's destination the desired intersection:

destination (a, changeDestination (a, n, e)) =
if n elementOf (connectedNodes (pos (a, e) e) then n
else error ("not a node")

The agent constructs knowledge about the environment while it moves. The operation *learnConnection* constructs the belief about the last segments traveled (start and end intersection and its length) and accumulates these beliefs about the environment. The operation *exploreEnv* lets an agent systematically travel all connections in the environment and accumulate complete knowledge about it. Agents can determine the shortest path (here simulated with the algorithm given by Dijkstra) to a destination based on their knowledge and move to a desired target following the planned path using *moveAlongPath* (using single steps of *moveOneTo).*

```
class Agents agent env where
            pos :: agent -> env -> Node
            destination :: agent -> env -> Node

            move :: agent -> env -> env
            changeDestination :: agent -> Node -> env -> env

            moveOneTowards :: agent -> Node -> env -> env
            learnConnection :: agent -> env -> env
            exploreEnv :: agent -> env -> env

            moveAlongPath :: [Node] -> agent -> env -> env
            pathFromTo :: agent -> Node -> Node -> env -> [Node]
            moveTo :: agent -> Node -> env -> env
```

Positions are tuples with a node and a coordinate pair; an edge connects two nodes with a certain cost, which is encoded as a real number. A possible data structure for agents contains the beliefs as a list of edges and position recordings, which are used only by map-makers:

```
data Agent = Agent AId Node Node [ConnectionCost] [Position]

data AId = AId Int deriving (Show, Eq)
type ConnectionCost = Edge Node

data Cost = Cost Float | CostMax
data Edge n = Edge n n Cost
```

An ordinary agent after having traveled over some segments has a knowledge, which is represented as (using the codes from Figure 4):

```
Agent A1 at Node 4 destination Node 2 beliefs
            Node 4 to Node 2 dist 3.20156
            Node 2 to Node 1 dist 1.41421
```

6.3.1 Map Making Agent. The map-making agent is a specialized agent and explores first the environment and then draws a map. In addition to the observation of connections, any agent is capable; it can observe the coordinate values of his current position. The map-maker can draw a map based on his current knowledge or can draw a sketch of a path between two nodes (used in section 9, Figure 10).

```
class MapMakers agent environment where
            isMapMaker :: agent -> environment -> Bool
            getCoords :: agent -> environment -> Vec2
            learnPos :: agent -> environment -> environment
            drawMap :: agent -> environment -> environment
            drawPathMap :: Node -> Node -> agent -> environment ->
environmentchange – is mapMaker
```

A map-making agent after having visited node 1,2 and 5 has also coordinates for these nodes (using again the codes from Figure 4):

```
Agent A1 at Node 5 destination Node 8 beliefs
            Node 8 to Node 5 dist 3.60555
            Node 3 to Node 5 dist 5.09902
            Node 5 to Node 2 dist 2.5
            Node 4 to Node 2 dist 3.20156
            Node 2 to Node 1 dist 1.41421
            Node 3 to Node 1 dist 3.20156
 visited
            Node 5:(5.0/8.0)
            Node 2:(3.0/6.5)
            Node 1:(2.0/5.5)
```

6.3.2 Map-Using Agents.
The map-using agents have the task of moving from the node they are located at to another node in the environment. Their locomotion operations are the same as for all agents. They intend to travel the shortest path (minimal distance). A map-user first reads the map (using *readMap*) and adds the knowledge acquired to his set of beliefs about the environment before he plans the shortest path to his destination node.

```
class MapUsers agent environment where
            readMap :: agent -> environment -> environment
```

6.4 Maps

Maps are artifacts, which exist in the environment (for simplicity, only one map is present in the model at any given time). The map-making agent produces the map usually after he has collected all beliefs about the environment. The map represents these beliefs in a (simulated) graphical format.

Maps are simulated in the model as a list of line segments (with start and end map coordinates) and labels at the intersection coordinates; one can think of this as suitable instructions for drawing a map with a computerized plotter. The map, in the form of the drawing instructions, is then read by the map-using agent and translated into a list of beliefs. This representation of the map avoids the need to simulate

drawing a bitmap and then using pattern recognition to analyze it; it leaves out the graphical restrictions of map-making and map reading.

Maps can be drawn and read, as well as sketches of a path (Figure 10):

```
class Maps aMap where
            drawTheMap :: [ConnectionCost:] -> [Position] -> aMap
            drawAPath :: [Node] -> [Position] -> aMap
            readTheMap :: aMap -> [ConnectionCost]
```

They are represented as

```
data Map = Map [Line] [Label]
data Line = Draw Vec2 Vec2
data Label = Label Node Vec2
```

7 Coding

The formalization has been coded in the functional notation of Haskell (Peyton Jones, Hughes et al. 1999). In a purely functional language, values cannot change and each movement of an agent is recorded as a new snapshot of the environment and not as a destructive change of the representation. This allows the use of standard mathematical logic, especially reasoning with substitution, and does not force to use temporal logic as would be necessary to reason with imperative programming languages.

Using a functional notation with some restrictions allows constructing executable prototypes (models for the abstract algebras constructed). These help to check that a formal system captures correctly our intentions. The following test starts with two agents “Jan” and “Dan” (more would be possible) in an environment with the streets from the center of Santa Barbara (with the coding shown in Figure 4). Jan is a “map-maker” and explores the environment. We can ask him for the path from Node 1 to Node 9 and get the shortest path. The same question to Dan gives no answer, as he has no knowledge yet. If Jan draws a map (*env2*) and Dan reads it (*env3*), then Dan can give the correct answer as well. This answer is the same as if Dan had explored the environment himself (*env1a*). The simulated system exhibits this behavior and confirms that our intuition about maps and the formalization correspond as explained in section 5. The following text shows a sequence of `code` and the *responses* from the system:

```
-- readable names for the agents:
jan = AId 1
dan = AId 2

-- create two agents at node 1 destination in direction of node 2

jan0 = Agent jan (Node 1) (Node 2) [] []
dan0 = Agent dan (Node 1) (Node 2) [] []


env0 = Env santaBarbara [jan0, dan0] emptyMap
--the two agents with the streets of Santa Barbara (figure 9)

env1' = learnPos jan env0
env1 = exploreEnv jan env1'
```

```
-- the positions of jan and dan
janpos1 = pos jan env1
danpos1 = pos dan env1
```

```
test input> janpos1
  Node 3
test input > danpos1
  Node 1
```

```
-- the path from 1 to 9
janpath1 = pathFromTo jan (Node 1) (Node 9) env1
danpath1 = pathFromTo dan (Node 1) (Node 9) env1
```

```
test input> janpath1
  [Node 1,Node 2,Node 4,Node 7,Node 9]
test input> danpath1
  []
```

```
-- jan draws map and dan reads it
env2 = drawMap jan env1
env3 = readMap dan env2

danpath3 = pathFromTo dan (Node 1) (Node 9) env3
```

```
test input> danpath3
  [Node 1,Node 2,Node 4,Node 7,Node 9]
```

```
-- this path is the same as if dan had explored the environment itself:

env1a = exploreEnv dan env0
danpath1a = pathFromTo dan (Node 1) (Node 9) env1a
env2a = drawPathMap (Node 1) (Node 9) jan env1
env3a = readMap dan env2a

danpath3a = pathFromTo dan (Node 1) (Node 9) env3a
```

```
test input> danpath3a
  [Node 1,Node 2,Node 4,Node 7,Node 9]
```

8 Definition of Correctness of a Map

In this environment, a formal and stringent definition for a map to be a correct representation of reality is possible. A map is correct if the result of an operation based on the information acquired from the map is the same as if the agent would have explored the world to gain the same information. The proof is in two steps: completeness and correctness. Completeness assures that all relevant elements – here nodes and segments – are transformed between the respective representations. Correctness requires that the transformations preserve the properties important for the decision (here the determination of the shortest path).

8.1 Completeness: Collecting All Observations into Beliefs

The operations to explore the environment and gradually learn about it or the exploration of a map are a repeated application of an operation *'learnConnection'*, which is

applied to all segments in the environment, respectively, the map. The construction of the beliefs of an agent about the environment can then be seen as a transformation between two data structures: the data structure which represents the environment is transformed into the internal structure of the beliefs. Similarly is the construction of the map a transformation between the data structure of the agent's beliefs into the list of drawing instructions; reading the map is the transformation of the data element of the map into beliefs.

We have to show that these transformations are applied to all elements and nothing is 'overlooked'. The *exploreEnv* operation is quite complex. It explores a node at a time, learning all segments, which start at this node, and keeps a list of all nodes ever seen. The environment is completely explored if all nodes where completely explored.

Drawing the map is a transformation procedure; coded with the second order function *map*, which applies a transformation to each element in a list. The transformation changes the belief into a drawing instruction. Reading the map is a similar function, taking line after line from the map and building a list of beliefs.

8.2 Correctness: Transformations Preserve the Important Properties

The different transformation for individual objects must preserve the properties necessary for the correct determination of the shortest path.

- A street segment is added to the beliefs after it is traveled; having traveled the segment ensures that the segment is viable and the cost is the cost just observed. Surveyors correctly observe the coordinate values for intersections.
- Map-makers translate each segment into a line drawn. The positions are based on the observed coordinate values for intersections.
- Map-users read the drawn line as viable segments and use the length of the line as an indication of the cost.

These operations guarantee that beliefs about viable street segments by the map-maker are communicated to the map-users. The (relative) cost is communicated correctly if the cost function is based on distance only. These transformations could be more realistic and include systematic and random errors and we could then observe the effects on the determination of the shortest path.

8.3 Discussion

In this example, where the observation and the use of the map are based on the same operation, nothing can go wrong. The model, however, indicates the potential for errors in communication. Here two examples:

8.3.1 Problems with the classification of elements. The world contains different classes of pathways, which can be driven, biked or walked, and not all segments can be passed with all vehicles. The classification of the road must be included in the map to allow use of the map for car drivers, bikers and persons walking. These problems seem trivial, but some of the current In-Car Navigation systems recommend paths, which include segments of a bike path!

In the simulation, if the exploring agent uses the same mode of locomotion as the map user, then correct communication is assured. If the exploring agent rides a (simulated) bike and the map using agent drives a (simulated) car, one may discover that the shortest path determined is using segments of a bike path the car driving agent cannot travel on or may find that a shorter route using an interstate highway is not found, because the map-making agent could not travel there and did not include it.

In general, the map-makers are not using the same operation that the map-user executes. The correctness of the map then depends on the composition of the transformation functions from observations of the map-maker to beliefs in the map user. The same criteria must be used during observation when the coding of an object is fixed. For example, while classifying roads using air photographs only road width, but not police regulations, are available to decide on the coding. This may classify some wide road segments which are closed for traffic as viable.

Users with different tasks may require different maps (or at least careful coding). A map for a hiker must be different from the map for driving – and indeed road maps for car driving are published separately from the maps for bikers or hiking maps. If a geographic database should be constructed for multiple purposes, then the properties which differentiate uses of objects must be recorded separately: the physical width and carrying capacity of a road must be recorded separately from the traffic regulations for the same road. It becomes then possible to establish the particular combinations of classifications, which simulate the intended type of use.

8.3.2 Problems with the transformation.

If the function to draw the map is using one of the many map projections, which do not preserve distances, then the representation of distances on the map is not representative of the distance between the nodes (but systematically distorted). The map-reader's naïve approach to link the distance between two nodes on the map with the cost for travel is then wrong and can lead to an error in determining the shortest path. More questions arise if the travel cost is a complex function of distance and other elements, e.g., the Swiss hiker's rule:

time (h) = distance(km)/5 + total ascent (m)/300 + total descent (m)/500

9. Effectiveness of Maps to Communicate Spatial Information

In this context, one may address the question why maps are so effective to communicate information about a complex environment in comparison to verbal descriptions. Take the small part of downtown Santa Barbara in Figure 4 and imagine communicating the information verbally: it would read as a long list, describing each segment, with the intersection it starts and ends:

The first segment of State St runs from Canon Perdido St to Ortega St, the next segment runs from Ortega St to Cota St. The first segment of Anacapa St runs from Canon Perdido St to Ortega St, etc., etc. This list contains a total of 12 segment descriptions, is tedious and does not communicate well. Alternatives would use the naming of 9 nodes and 24 incidence relations.

For areas where streets are regularly laid out, abbreviations could be invented. For example, in large parts of Santa Barbara, it is sufficient to know which streets run (conventionally) North-South and which East-West and to know the order in which they are encountered. This does not work for areas where the street network is irregular and a detailed description, for example, for areas, where an Interstate highway or a railway line intersect and distort the regular grid.

A verbal description for a street network is tedious and verbose, because it must create communicable identifiers for each object; for example, a name must be given to each intersection, such that another street segment starting or ending at the same location can refer to it. A graphical communication uses the spatial location to create references for the locations and does not need other names. The incidence is expressed as spatial position on paper and picked up by the eye. The information retained is the same, but the communication is more direct, using the visual channel. It is curious to note that American Sign Language, which is a well-documented natural language, uses a similar device of 'location' used as references. The speaker may designate a location in the (signing) space before him to stand for a person or place he will later refer to. A later reference to this person or place is then made by simply pointing to the designated location, using the location as a reference to the objects (Emmorey 1996).

The situation is different when only a specific path should be communicated. The list of instructions is shorter and simpler than the sketch (Figure 10). The instructions for a path from Borders (Intersection Canon Perdido St and State St) to Playa Azul (Intersection of Santa Barbara St with Cota St):

Follow Canon Perdido Street to the East for one block,
Turn right and follow Anacapa Street for two blocks
Follow Cota St to the East for one block

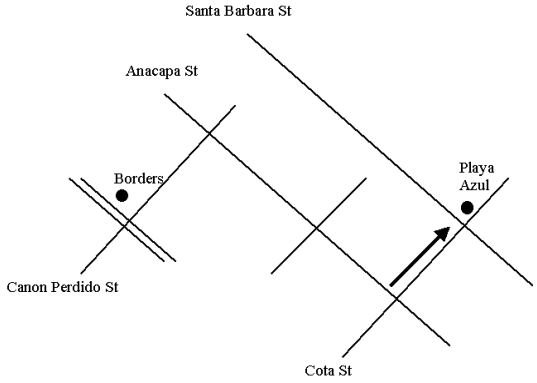

Fig. 10. Sketch for path from Borders to Playa Azul

In the language of the agents, a list of nodes as the shortest path is communicate as:

```
[Node 1,Node 2,Node 4,Node 7,Node 9]
```

Each of the representations is short and can be communicated using a linear channel, e.g., verbally). A sketch map would be somewhat more complex as the following example of a simulated map demonstrates

```
env4 = drawPathMap (Node 1) (Node 9) jan env1

line:(2.0/5.5), (3.0/6.5),line:(3.0/6.5), (5.0/4.0),
line:(5.0/4.0), (6.0/3.0),line:(6.0/3.0), (8.0/4.0),
label Node 4 (5.0/4.0),label Node 7 (6.0/3.0),label Node 9 (8.0/4.0),
label Node 2 (3.0/6.5),label Node 1 (2.0/5.5),
```

10. Conclusion

A framework for the formalization of the production and use of maps and cartographic diagram is described. The model is two-tiered; it contains a representation of what stands for *reality* and what stands for the *beliefs* of multiple, simulated agents about reality. The model is natural, allows misrepresentation and is fine-grained. It goes beyond current models, as it permits to model the observation processes of the agents and the agents' actions, which use the information collected. It can include errors in these processes or in the information stored.

The production and use of maps or diagrams for navigation can be described in this computational model, which includes processes for exploring the environment while traveling, casting the information collected by the agent into a graphical form, which can be communicated to and be used by another agent. The semantics of the map is directly related to the processes that observe reality or use the data. Correctness of a map can be established in this formalization. It is directly related to the connection between the operations used for observing and representing reality and the operations the map users intend to perform.

Using an executable functional language to construct multi-agent models allows experimenting. The code is very compact and takes only 5 pages, plus 3 pages for the graph related utilities, including the shortest path algorithm. The code is available from http://www.geoinfo.tuwien.ac.at.

In this multi-agent framework, other related questions can be explored. For example, the effects of incomplete street maps on navigation can be simulated and tested, how much longer the path traveled becomes and what are the best strategies for users to cope with the incomplete information. One can also explore different strategies for map users to deal with observed differences between the map and the environment.

Acknowledgements

This contribution is based on research I carried out at the University of California, Santa Barbara. I thank both the National Center of Geographic Information and

Analysis, and the Department of Geography for the hospitality and the support they have provided. I have greatly benefited from the discussion with the students in the course 'Formalization for GIS' I taught here. Discussions with Dan Montello, Helen Couclelis, Mike Goodchild, Waldo Tobler, and Jordan Hastings were invaluable to sharpen my arguments.

I also thank Werner Kuhn, Martin Raubal, Hartwig Hochmair, Damir Medak and Annette von Wolff for the patience to listen to and comment on these ideas in various stages of progression. The contribution to help clarify my ideas made by Haskell is invaluable, and I thank the Haskell community, especially Mark Jones, for the Hugs implementation. The comments from two unknown reviewers were extremely useful for the revision of the paper. Roswitha Markwart copy-edited the text and Hartwig Hochmair improved the figures.

References

Abelson, H. and A. A. Disessa (1986). Turtle Geometry : The Computer As a Medium for Exploring Mathematics. Cambridge, Mass., MIT Press.

Baeijs, C., Y. Demazeau, et al. (1995). SIGMA: Approche multi-agents pour la generalisation cartographique. CASSINI'95, Marseille, CNRS.

Braitenberg, V. (1984). Vehicles, experiments in synthetic psychology. Cambridge, MA, MIT Press.

Burrough, P. A. and A. U. Frank (1995). "Concepts and paradigms in spatial information: Are current geographic information systems truly generic?" International Journal of Geographical Information Systems **9**(2): 101-116.

Burrough, P. A. and A. U. Frank, Eds. (1996). Geographic Objects with Indeterminate Boundaries. GISDATA Series. London, Taylor & Francis.

Burrough, P. A. W. v. D. and G. Heuvelink (1988). Linking Spatial Process Models and GIS: A Marriage of Convenience or a Blossoming Partnership? GIS/LIS'88, Third Annual International Conference, San Antonio, Texas, ACSM, ASPRS, AAG, URISA.

Corbett, J. (1975). Topological Principles in Cartography. 2nd International Symposium on Computer-Assisted Cartography, Reston, VA.

Corbett, J. P. (1979). Topological Principles of Cartography, Bureau of the Census, US Department of Commerce.

Davis, E. (1990). Representation of Commonsense Knowledge. San Mateo, CA, Morgan Kaufmann Publishers, Inc.

Dijkstra, E. W. (1959). "A note on two problems in connection with graphs." Numerische Mathematik (1): 269-271.

Dreyfuss (1998). What Computers Still Cannot Do. Cambridge, Mass., The MIT Press.

Emmorey, K. (1996). The cofluence of space and language in signed language. Language and Space. P. Bloom, M. A. Peterson, L. Nadel and M. F. Garett. Cambridge, Mass., MIT Press**:** 171 - 210.

Epstein, J. M. and R. Axtell (1996). Growing Artificial Societies. Washington, D.C., Brookings Institution Press.

Ferber, J., Ed. (1998). Multi-Agent Systems - An Introduction to Distributed Artificial Intelligence, Addison-Wesley.

Frank, A. U. (1998). GIS for Politics. GIS Planet'98, Lisbon, Portugal (September 9-11, 1998), IMERSIV.

Hudak, P., S. L. Peyton Jones, et al. (1992). "Report on the functional Programming Language Haskell, Version 1.2." ACM SIGPLAN Notices **27**(5): 1-164.

Johnson, M. (1987). The Body in the Mind: The Bodily Basis of Meaning, Imagination, and Reason. Chicago, University of Chicago Press.

Kirschenhofer, P. (1995). The Mathematical Foundation of Graphs and Topolo gy for GIS. Geographic Information Systems - Material for a Post Graduate Course. A. U. Frank. Vienna, Department of Geoinformation, TU Vienna. **1:** 155-176.

Kuipers, B. and T. S. Levitt (1990). Navigation and Mapping in Large-Scale Space. Advances in Spatial Reasoning. S.-s. Chen. Norwood, NJ, Ablex Publishing Corp. **2:** 207 - 251.

Kuipers, B. J. and T. S. Levitt (1978). "Navigation and mapping in large-scale space." AI Magazine **9**(2): 25-43.

Lakoff, G. and M. Johnson (1999). Philosophy in the Flesh. New York, Basic books.

Lechthaler, M. (1999). "Merkmale der Datenqualitaet im Kartographischen Modellbildungsprozess." Kartographische Nachrichten **49**(6): 241-245.

Papert, S. and J. Sculley (1980). Mindstorms: Children, Computers and Powerful Ideas. New York, Basic Books.

Peterson, J., K. Hammond, et al. (1997). "The Haskell 1.4 Report." http://haskell.org/report/index.html.

Peyton Jones, S., J. Hughes, et al. (1999). Haskell 98: A Non-strict, Purely Functional Language.

Pierce, B. C. (1993). Basic Category Theory for Computer Scientists. Cambridge, Mass., MIT Press.

Pinker, S. (1997). How the Mind Works. New York, W. W. Norton.

Stich, S. P. and T. A. Warfield, Eds. (1994). Mental Representation. Cambridge, Mass., Basil Blackwell.

Tchen, Y.-S. (1967). Je parle Chinois. Paris, Librairie d'Amerique et d'Orient.

Walters, R. F. C. (1991). Categories and computer science. Cambridge, UK, Carslaw Publications.

Weibel, R. (1995). "Map generalization in the context of digital systems." CaGIS **22**(4): 259-263.

Weiss, G. (1999). Multi-Agent Systems: A Modern Approach to Distributed Artificial Intelligence. Cambridge, Mass., The MIT Press.