

Granularity Transformations in Wayfinding

Sabine Timpf¹ and Werner Kuhn²

¹Department of Geography
University of Zurich
timpf@geo.unizh.ch

²Institute for Geoinformatics
University of Muenster
kuhn@ifgi.uni-muenster.de

Abstract. Wayfinding in road networks is a hierarchical process. It involves a sequence of tasks, starting with route planning, continuing with the extraction of wayfinding instructions, and leading to the actual driving. From one task level to the next, the relevant road network becomes more detailed. How does the wayfinding process change? Building on a previous, informal hierarchical highway navigation model and on graph granulation theory, we are working toward a theory of granularity transformations for wayfinding processes. The paper shows the first results: a formal ontology of wayfinding at the planning level and an informal model of granularity mappings.

Keywords: vehicle navigation, wayfinding, hierarchies, activity theory, graph granulation.

1 Introduction

Graph granulation theory [12] is neutral with respect to the choice of graph elements at a particular granularity level. This choice has to be guided by domain models, leading to application-specific network ontologies. A minimal ontology of road networks can be derived from a formalization of wayfinding activities at each level [9]. This idea is being applied here to a formalization of Timpf's hierarchical highway navigation process model [16].

Human beings use several conceptual models for different parts of geographic space to carry out a single navigation task. Different tasks require different models of space, often using different levels of detail. Each task is represented in a conceptual model and all models together form a cognitive map or collage for navigation. The different models of space need to be processed simultaneously or in succession to completely carry out a navigation task. Humans are very good at that type of reasoning and switch without great effort from one model of space to another. Today's computational navigation systems, on the other hand, cannot deal well with multiple representations and task mappings between them.

Existing spatial data hierarchies refine objects and operations from one level to the next, but the objects and operations essentially stay the same across the levels [1].

Task hierarchies, by contrast, refine the tasks from one level to the next and the objects and operations change with the level [7].

Timpf's cognitive architecture of interstate navigation [16] consists of three distinct conceptual models (levels): planning, instructing, and driving. Each level is characterized by a function computing information about a route. Each function takes the result of its predecessor and computes the route in the road network at its level. Thus, a concatenation of these wayfinding functions leads from the origin and destination of a trip all the way to detailed driving behavior.

The purpose of our work in progress is to gain a better understanding of these wayfinding functions and of the granularity mappings they induce on road networks. We present an executable formal specification of selected navigation functions in the functional language Haskell, specifically in its HUGS dialect [11]. Functional languages have great appeal for software engineering, because algebraic specifications [8] can be written and tested in them [6]. By the same token, they serve as test beds for formal algebraic theories.

The paper presents methodological and domain-specific results. Methodologically, we show that the use of functional languages for ontologies goes beyond collections of abstract data type specifications to comprehensive, layered object *and task* models with mappings among them. For the domain of highway navigation, we present the first (planning) level of a formal hierarchical task model.

The results are of interest to the spatial reasoning, geographic information science, and cognitive science communities. They touch on general issues in navigation and wayfinding, hierarchical reasoning, and formal ontological modeling. In practice, such formalizations of wayfinding can be used as algebraic specifications for more sophisticated navigation systems [17], supporting the planning, instructing, and driving processes on highway networks.

The remainder of the paper is structured as follows: section 2 presents the conceptual model for the three levels of wayfinding tasks; section 3 shows the granularity mappings among task levels by examples; section 5 presents the formalization approach; and section 6 discusses the results and future work.

2 The Conceptual Model

A series of complex cognitive operations are involved in planning and executing a journey on a highway network. Some operations are more general at one level (e.g., take exit) and are broken down into several operations at another level (e.g., change to appropriate lane for exiting, take ramp). There are operations that require bodily actions in one level but do not affect any other level (e.g., accelerating the vehicle). One of the authors has previously structured highway navigation operations into three task levels: planning, instructing, and driving (Timpf, et al. 1992). Table 1 shows how they organize the activity of wayfinding on highways and how they can be further subdivided into operations.

Table 1. Informal task ontology of wayfinding

Activity	Wayfinding: get from A to B		
Task Levels	<i>Plan</i> make a plan	<i>Instruct</i> produce instructions	<i>Drive</i> carry out instructions
Operations	find routes, determine constraints	take entrance, follow highway, change highway, take exit	take onRamp, change lane, change speed, proceed to, take offRamp

The planning, instructing, and driving tasks operate in different spatial domains. Planning involves knowledge about the places where one is and where one wants to go, as well as relevant additional places in between and the overall highway network containing them. The instruction task involves knowledge about the decision points along the route resulting from the planning task. The driving task needs information about when to drive where and how, but also introduces a body of actions of its own (e.g., change lane).

2.1 Ontologies at the Three Task Levels

At the Planning Level objects of the following types exist: *Place*, *Highway*, and *PLGraph* (the highway network at this level). The origin and destination of a trip are instances of places. Fig.1 shows an excerpt from the US highway network, labeled with place and highway names.

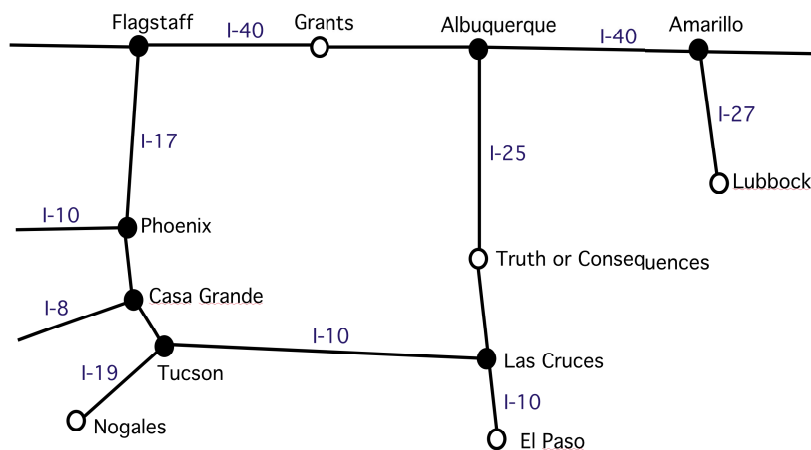


Fig. 1. Objects at the Planning Level (for a part of the US highway network located in New Mexico)

The *Instructional Level* (Fig. 2) introduces objects of type *Entrance*, *Exit*, *Section*, *Junction*, and *ILGraph* (the highway network at this level). A *Section* leads from an entrance to an exit on the same highway, while a *Junction* connects an exit to an entrance on another highway.

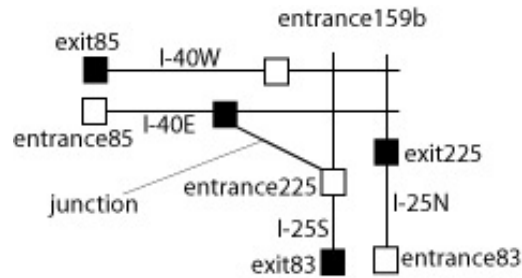


Fig. 2. Objects at the Instructional Level

The *Driving Level* (Fig. 3) is the most detailed, containing the objects and operations necessary to drive a vehicle with the instructions gotten at the previous level. Its pertinent objects are *lanes*, *ramps*, and the *DLGraph* (the highway network at this level). Three kinds of lanes exist: *travel*, *passing*, and *breakdown lanes*. *OnRamps* lead onto a highway, while *offRamps* leave a highway.

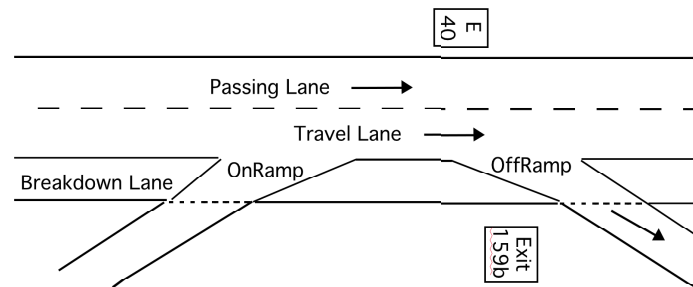


Fig. 3. Objects at the Driving Level

2.2 Graph Model

The objects at the Planning, Instructional, and Driving Levels are best represented by graphs and their parts (Fig. 4). The graph at the Planning Level contains places as nodes; highways are represented by named sequences of nodes connected by undirected edges. At the Instructional Level, nodes stand for exits and entrances, while directed edges represent highway sections and junctions. At the Driving Level, nodes represent ramps and (directed) edges represent lanes.

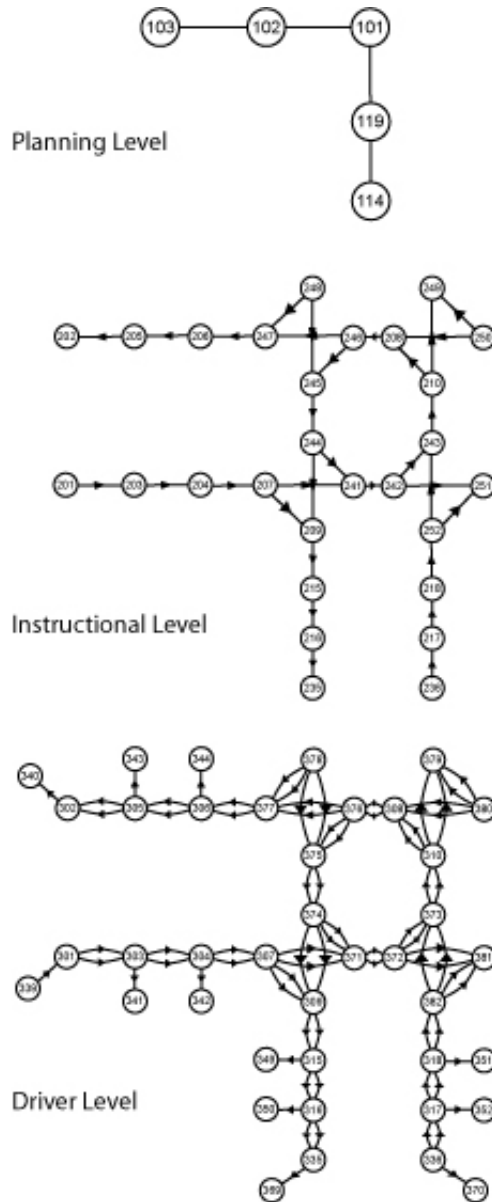


Fig. 4. Graphs representing the highway network at the three levels of detail

2.3 Reasoning in the Network: Navigation Tasks

Spatial reasoning in a highway network is a top-down process. Given the origin and destination of a trip on a highway network, reasoning at the Planning Level returns a *plan*. This plan is a list of places and highways necessary to get from the origin to the

destination, passing through highway interchanges. It is fed into the Instructional Level, which produces a list of *instructions*. These in turn are used as input to the Driving Level, which transforms them into driving *actions*.

The major operation at the Planning Level is to find a path from the origin to the destination. This path (which is a sequence of places to travel through) is then expressed as a sequence of place and highway names; e.g., (<Grants, I-40>, <Albuquerque, I-25>, <Truth or Consequences, reached>).

Table 2. Reasoning at the Planning Level

<i>plan (origin, destination, piGraph)</i>	
reasoning chain:	(<origin, HighwayName>, <Place, HighwayName>, ... <destination, "reached">)

The major operation at the *Instructional Level* (Table 3) is to produce instructions for a given plan. Information on the direction of the highway sections is taken from the highway network at this level, producing a sequence of triples <HighwayName, HighwayDirection, Distance>. The reasoning chain starts with finding the entrance, then taking it, and following the first highway to the first relevant interchange. This is repeated for all highways in the plan, followed by taking the exit at the destination place.

Table 3. Reasoning at the Instructional Level

<i>instructions (plan, iiGraph)</i>	
reasoning chain:	take_entrance (origin, firstHighway), follow <HighwayName, HighwayDirection, Distance> to interchange change_at junction, follow <HighwayName, HighwayDirection, Distance > to interchange ... take_exit at destination.

The operations at the *Driving Level* (Table 4) involve the actions to get from the *origin* to the *destination* with the help of the instructions. The *onRamp* brings one onto the acceleration lane, where it is necessary to *accelerate* and then to *change lane* to the *left* before being on the *highway*. Then, one *follows* the *highway* until the sign with the interchange mentioned in the instructions comes up and actions are required again. Then one has to *changeover* to the rightmost lane to be able to exit, an action composed of *decelerating* and taking the *offRamp*. In case of a *junction*, the driver will *proceed* to the next *highway* and accelerate again.

Table 4. Reasoning at the Driving Level

<i>drive (instructions, dlGraph)</i>	
reasoning chain:	<pre> take_OnRamp (firstHighwaySection, firstDirection), accelerate, change_lane(left), follow < HighwayName, HighwayDirection, Distance> to (landmark), change_lane(right) until (rightneighbor(lane) = BreakdownLane), decelerate, take_OffRamp <HighwaySection, Direction>, proceed < HighwaySection, Direction>, accelerate,, take_OffRamp <HighwaySection, destination> </pre>

At this level, it is assumed that the driver knows how to steer a vehicle, how to accelerate or how to proceed. These actions are not further broken down. It is also assumed that the driver knows how to handle the car in the presence of other cars.

3 Granularity Mappings

Graph granulation theory posits two operations for graph simplification: *selection* and *amalgamation* (Stell and Worboys, 1999). Selection retains the nodes from one level that will be represented at a level of less detail. Any non-selected nodes disappear at the coarser level of detail. Amalgamation maps some paths at one level to nodes at a coarser level of detail.

Amalgamation is the simplification operation among our three levels of granulation. For example, in Fig. 4, the path leading from node 339 to 301 at the Driving Level is collapsed to node 201 at the Instructional Level. We have identified four different types of amalgamation in highway networks:

- Path -> Node
- Path -> (simplified)Path
- connected sub-graph -> node
- multi-Edge -> single Edge

The mappings between the complete graph at a level and the corresponding path as well as between paths and routes are selections (Fig. 5). The selection process leading from paths to routes (when each is seen as a graph) is exactly the selection operation of graph granulation theory. The selection process leading from the complete graphs to paths is a special selection operation producing a sub-graph of the original graph.

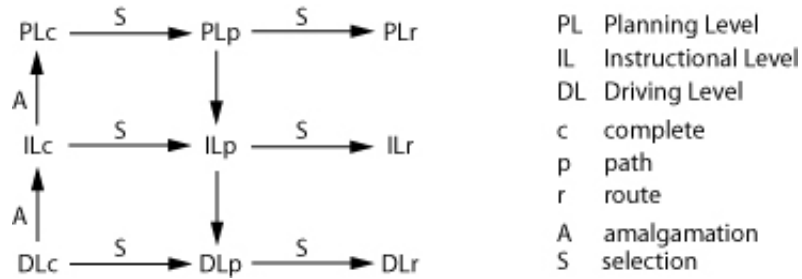


Fig. 5. Mappings

The correspondences between objects at different levels, resulting from the amalgamations, are shown in Table 5. They represent a special case of the aggregation and generalization hierarchies defined for graph objects in ([14]; [13]). Object types that are not explicit parts of the ontologies are put in parentheses (for example, individual edges do not play a role at the Planning Level).

Table 5. Corresponding objects

	Graph	Path	Edge	Node
Planning Level	PLGraph	Highway Route	(HwSegment)	Place (Interchange)
Instructional Level	ILGraph	Route	Section Junction	Exit Entrance
Driving Level	DLGraph	Route	Lane	OnRamp OffRamp

Our goal is a hierarchical graph structure that represents these amalgamations. Since the actual mappings are different for each instance (e.g., places can contain various combinations of exits and entrances, linked by sections and junctions; sections and junctions may consist of any number of lanes), this structure can only be described extensionally. Graph granulation theory (Stell and Worboys, 1999) proposes simplification graphs to represent the composition of each higher-level object from lower-level objects explicitly.

4 Formalization

We formalize our navigation model in HUGS, a dialect of the functional programming language Haskell [11]. Functional specifications serve as a workbench on which theories (e.g., of spatial cognition) can be

- worked out concisely and with formal algebraic rigor,
- tested for correctness (with respect to the requirements),
- adapted to ontological commitments

- compared and
- combined with each other [4].

Functional languages also have a great appeal for software engineering, because algebraic specifications [8] can be written and tested in them [6]. In this context, they combine the benefits of

- clean semantics (in particular, referential transparency for equational reasoning as well as a clean multiple inheritance concept),
- executability (allowing software engineers to test what they specify), and
- higher order capabilities (leading to leaner, more elegant descriptions).

Encouraged by a series of successful applications to non-trivial software engineering tasks ([2], [5], [15]), we have used functional languages for ontological research into the structure of application domains ([10]; Frank 1999; Kuhn 2001). The work presented here continues on this path by formalizing hierarchical navigation tasks on highway networks.

The object classes of the data model (i.e., the Haskell data types) are based on notions of graph theory. For instance, a highway section is an edge between nodes in the highway network at the Instructional Level. We are using Erwig's inductive graph library [3] to supply the necessary graph data types and algorithms. The HUGS code below also uses some elementary list functions. Parentheses construct tuples (specifically, pairs), and brackets construct lists.

The data type definitions formalize the ontologies for each task level:

```
-- Planning Level object types
type Place = Node
type Highway = (HighwayName, [Place])
type PLGraph = Graph PlaceName EdgeLength
type Route = Path
type Plan = [(PlaceName, HighwayName)]

-- Planning Level attributes and auxiliary types
type PlaceName = Name
type HighwayName = Name
type Highways = [Highway]
type Leg = (Place, Highway)
type Legs = [Leg]

-- Instruction Level object types
type Entrance = Node
type Exit = Node
type Section = Edge
type Junction = Edge
type ILGraph = Graph EName EdgeLength

-- Instruction Level attributes and auxiliary types
type EName = Name

-- Driving Level object types
```

```

type Ramp = Edge
type Lane = Edge
type DLGraph = Graph Name EdgeLength

```

At the planning level, the route from origin to destination is determined by the shortest path operation (`sp`) applied to the highway graph at this level (`PLGraph`).

```

route :: Place -> Place -> PLGraph -> Route
route origin destination plg = sp origin destination plg

```

This route is a path in the graph, i.e., a sequence of nodes. It has to be translated into a plan, i.e., a sequence of pairs with names of places and highways to follow. For this purpose, information about the highways has to be combined with the route and the graph. This is done in a function computing the “legs” (a sequence of pairs of places and highways) leading from origin to destination:

```

legs :: Route -> Highways -> Legs
legs (x:[]) hws = [(x, endHighway)]
legs rt hws = (head rt, firstHighway rt hws) : legs (tail rt)
hws

```

The recursively applied function `firstHighway` computes the first highway to take on a route:

```

firstHighway :: Route -> Highways -> Highway
firstHighway rt hws = fromJust (find (hwConnects (rt !! 0)
(rt !! 1)) hws)

```

The first highway is determined by finding, among all highways, the highway that connects the first and second place (assuming there is only one):

```

hwConnects :: Place -> Place -> Highway -> Bool
hwConnects p1 p2 hw = (elem p1 (snd hw)) && (elem p2 (snd
hw))

```

From the legs of the trip, those legs which continue on the same highway can be eliminated:

```

planModel :: Legs -> Legs
planModel lgs = map head (groupBy sameHighway lgs)

sameHighway :: Leg -> Leg -> Bool
sameHighway (p1, hw1) (p2, hw2) = hw1 == hw2

```

Finally, this (internal) model of a plan is translated into an (external) view expressing it by the names of places and interchanges:

```

planView :: Legs -> PLGraph -> Plan
planView (x:[]) plg = [(placeName (fst x) plg, fst (snd x))]
planView pm plg = (placeName (fst (head pm)) plg, fst (snd
(head pm))) : (planView (tail pm) plg)

```

This completes the formal model at the Planning Level. The given HUGS code allows for the computation of trip plans on any highway network that is expressed as an inductive graph.

At the Instructional Level, the `planModel` will get expanded into a list of highway entrances, segments, junctions, and exits, using the amalgamation functions to be

defined. Similarly, at the Driving Level, these instructions will be expanded into driving actions consisting of ramps and lanes to take.

5 Conclusions

Human beings use information at multiple levels of detail when navigating highway networks. This paper describes a conceptual model of the U.S. Interstate Network at three levels of reasoning: planning, instructing, and driving. The apparently simple everyday problem of navigating a highway network has been shown to contain a high degree of structure and complexity. Executable algebraic specifications and graph granulation theory have been applied to formalize this structure and test the results.

The formalization presented in this paper covers the first level of reasoning (planning tasks). It provides a framework for comparing the reasoning at the three levels. While planning involves the computation of a shortest path, finding instructions and transforming them to driving actions use granulation relationships between graphs, rather than graph operations at a single level. The definition of and interaction between the three levels is intended to provide a cognitively plausible model of actual human wayfinding processes within the U.S. Interstate Highway Network. We proposed objects and actions corresponding to the *physical* structure at each level and playing a role in real wayfinding processes.

The formal model can serve as a software specification (specifically, as the essential and abstract model) for navigation systems used for Interstate travel. Software for navigation systems is currently very limited in its support for hierarchical reasoning. The key benefits of choosing a functional language to write algebraic specifications for navigation operations are that the specified models can be tested and are semantically unambiguous.

Acknowledgments

The work reported here was supported by the University of Zürich, the University of Münster, and the Technical University of Vienna.

References

- [1] Car, A. (1997). Hierarchical Spatial Reasoning: Theoretical Consideration and its Application to Modeling Wayfinding. GeoInfo Series Vol. 10. TU Vienna: Dept. of Geoinformation.
- [2] Car, A. and A. U. Frank (1995). Formalization of Conceptual Models for GIS using Gofer. *Computers, Environment, and Urban Systems* 19(2): 89-98.
- [3] Erwig, M. (2001). Inductive Graphs and Functional Graph Algorithms. *Journal for Functional Programming* 11(5): 467-492.
- [4] Frank, A. U. (1999). One step up the abstraction ladder: Combining algebras - From functional pieces to a whole. *Spatial Information Theory*. C. Freksa and D. Mark, Springer-Verlag. Lecture Notes in Computer Science 1661.

- [5] Frank, A. U. and W. Kuhn (1995). Specifying Open GIS with Functional Languages. *Advances in Spatial Databases - 4th Internat. Symposium on Large Spatial Databases, SSD'95 (Portland, ME)*. M. Egenhofer and J. Herring. New York, Springer-Verlag: 184-195.
- [6] Frank, A. U. and W. Kuhn (1999). A Specification Language for Interoperable GIS. *Interoperating Geographic Information Systems*. M. F. Goodchild et al., Kluwer: 123-132.
- [7] Freksa, C. (1991). Qualitative Spatial Reasoning. In D. M. Mark & A. U. Frank (Eds.), *Cognitive and Linguistic Aspects of Geographic Space*. Dordrecht, The Netherlands: Kluwer Academic Press: 361-372.
- [8] Guttag, J. V. (1977). Abstract Data Types and the Development of Data Structures. *ACM Communications* 20(6): 396-404.
- [9] Kuhn, W., 2001. Ontologies in support of activities in geographical space. *International Journal of Geographical Information Science*, 15(7): 613-631.
- [10] Medak, D. (1997). Lifestyles - A Formal Model. Chorochronos Intensive Workshop '97, Petronell-Carnuntum, Austria, Dept. of Geoinformation, TU Vienna.
- [11] Peterson, J., K. Hammond, et al. (1997). The Haskell 1.4 Report. <http://haskell.org/report/index.html>.
- [12] Stell, J. G., & Worboys, M. F. (1999). Generalizing Graphs using amalgamation and selection. In R. H. Gueting & D. Papadias & F. Lochovsky (Eds.), *Advances in Spatial Databases, 6th Symposium, SSD'99* (Vol. 1651 LNCS, pp. 19-32): Springer.
- [13] Timpf, S. (1999). Abstraction, levels of detail, and hierarchies in map series. *Spatial Information Theory - cognitive and computational foundations of geographic information science*. C. Freksa and D.M. Mark. Berlin-Heidelberg, Springer-Verlag. Lecture Notes in Computer Science 1661: 125-140.
- [14] Timpf, S. (1998). Hierarchical structures in map series. GeoInfo Series Vol. 13. Vienna: Technical University Vienna.
- [15] Timpf, S. and A. U. Frank (1997). Using Hierarchical Spatial Data Structures for Hierarchical Spatial Reasoning. *Spatial Information Theory - A Theoretical Basis for GIS (International Conference COSIT'97)*. S. C. Hirtle and A. U. Frank. Berlin-Heidelberg, Springer-Verlag. Lecture Notes in Computer Science 1329: 69-83.
- [16] Timpf, S., G. S. Volta, et al. (1992). A Conceptual Model of Wayfinding Using Multiple Levels of Abstractions. *Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*. A. U. Frank, I. Campari and U. Formentini. Lecture Notes in Computer Science 639: 348-367.
- [17] White, M. (1991). Car navigation systems. *Geographical Information Systems: principles and applications*. D. J. Maguire, M. F. Goodchild and D. W. Rhind. Essex, Longman Scientific & Technical. 2: 115-125.