A1-D2

Geospatial Reasoning: Basic Concepts and Theory

Project title:	Reasoning on the Web with Rules and Semantics
Project acronym:	REWERSE
Project number:	IST-2004-506779
Project instrument:	EU FP6 Network of Excellence (NoE)
Project thematic priority:	Priority 2: Information Society Technologies (IST)
Document type:	D (deliverable)
Nature of document:	R (report)
Dissemination level:	PU (public)
Document number:	IST506779/Munich/A1-D2/D/PU/a1
Responsible editors:	H.J. Ohlbach
Reviewers:	
Contributing participants:	Munich
Contributing workpackages:	A1
Contractual date of deliverable:	31 March 2005
Actual submission date:	31 March 2005

Abstract
'Geospacial Reasoning' is spacial reasoning including geographic data. This is an extremely board area. In this deliverable we presented our approaches to cover enough aspects from this area such that it is useful for Semantic Web applications. An important building block is a hierarchy of graphs which combines very low level coordinate based computations with abstract symbolic reasoning. Another building block is the ontology of transport networks OTN. The graphs and the corresponding algorithms, together with OTN, provide the built-ins for the specification language MPLL. MPLL is a functional programming language, which, however, is mainly used to control the application of built-in algorithms, for example shortest path computations. With MPLL one can define customized locational notions, and evaluate them over the given data (maps, transport networks, user context etc.). The processing of uncertainty is certainly a major issue also for locational reasoning. So far we have investigated the use of fuzzy sets in this context. The FuTIRe library, which was developed for fuzzy temporal intervals can be used for one-dimensional fuzzy distributions in a higher dimensional space as well. It is easy to integrate the processing of two-dimensional fuzzy distributions which are only represented by their membership functions. This is sufficient as a first approximation to fuzzy reasoning in

two dimensions.

Keyword List
semantic web, geospatial notions, geospatial reasoning techniques

Project co-funded by the European Commission and the Swiss Federal Office for Education and Science within the Sixth Framework Programme.

Geospatial Reasoning: Basic Concepts and Theory

Hans Jürgen Ohlbach[1], **Bernhard Lorenz**[2]

[1] Department of Computer Science, University of Munich
Email: `ohlbach@pms.ifi.lmu.de`

[2] Department of Computer Science, University of Munich
Email: `lorenz@pms.ifi.lmu.de`

31 March 2005

Abstract
'Geospacial Reasoning' is spacial reasoning including geographic data. This is an extremely board area. In this deliverable we presented our approaches to cover enough aspects from this area such that it is useful for Semantic Web applications. An important building block is a hierarchy of graphs which combines very low level coordinate based computations with abstract symbolic reasoning. Another building block is the ontology of transport networks OTN. The graphs and the corresponding algorithms, together with OTN, provide the built-ins for the specification language MPLL. MPLL is a functional programming language, which, however, is mainly used to control the application of built-in algorithms, for example shortest path computations. With MPLL one can define customized locational notions, and evaluate them over the given data (maps, transport networks, user context etc.). The processing of uncertainty is certainly a major issue also for locational reasoning. So far we have investigated the use of fuzzy sets in this context. The FuTIRe library, which was developed for fuzzy temporal intervals can be used for one-dimensional fuzzy distributions in a higher dimensional space as well. It is easy to integrate the processing of two-dimensional fuzzy distributions which are only represented by their membership functions. This is sufficient as a first approximation to fuzzy reasoning in two dimensions.

Keyword List
semantic web, geospatial notions, geospatial reasoning techniques

Contents

1 Introduction and Motivation

The notion "geospatial reasoning" has an extremely broad interpretation. It covers simple coordinate computations via triangulation. It also covers route planning for all kinds of vehicles, path finding in networks, and even robot navigation. On the very abstract end of this spectrum we have pure logic based reasoning about spatial relations.

Since the world wide web, and therefore even more the semantic web can contain information about really everything of interest for human beings, all this, and much more, are interesting aspects of "geospatial reasoning" in the context of the semantic web. For a small research project, this is of course much too broad. Therefore we have to narrow down the the focus of the "geospatial reasoning" research in the A1 working group.

The focus of a research project can be narrowed down by considering only certain use cases, or even just one single use case. The use case in A1 is: *querying a relational or XML database with intelligent processing of locational information.* The database may contain pure locational information, for example the road map of Munich. It may, however, also contain other kinds of data, which is annotated with locational information, addresses, coordinates, spatial relations to other objects etc. The query may contain "locational" constraints which must be met by the answers. A query may, for example, be "give me all cinemas in the south of Munich". The problem here is to match *in the south of* with concrete addresses.

If we put no restrictions on the database and the queries this use case does in fact not restrict the general problem of "locational reasoning". Nevertheless, it illustrates the dimension of the problem and it helps to identify manageable subproblems.

1.1 Examples for Data and Queries

We illustrate the problem with a few typical examples. The examples cover by no means the whole area, but they give a first impression of the problems to be solved.

Example 1.1 Suppose we have some data about cities, states and countries. Entries could be:
(1) San Francisco is a city
(2) San Francisco is in California
(3) San Francisco has 3 million inhabitants
(4) California is in the USA.

A query could be: "give me all metropolises in the USA". In order to evaluate this query we need to:

- formulate the database entries in a logic based knowledge representation language, for example OWL or its underlying Description Logic.
- define the concept "metropolis" in the same knowledge representation language, e.g.

  $$metropolis = city \wedge atleast\ 1000000\ has\_inhabitant \tag{1}$$

  (A a metropolis is a city with at least 1 million inhabitants.)
- make a so called *instance test* for the database entries. The instance test would conclude from (2) and (4) that San Francisco is in the USA, and from (1) and (3) that San Francisco is a metropolis.

■

Example 1.2 Suppose the database contains the yellow pages entries, i.e. businesses with their addresses. A query could be: "give me the nearest pharmacy", with the context information that I am at a particular location X in the city, and with all the other context information about my current situation (availability of a car, luggage, my age and gender etc.).

This query could be evaluated in a naive way by selecting the pharmacy with the smallest geographic distances between it and the location X. This might be a first approximation, but it can give completely useless results. A pharmacy which is located very close by, but unfortunately it is on the other side of the river, and the next bridge is miles away, may not be a good choice.

The answers would be much more appropriate if we use, instead of the geographic distance, a metric which is determined by the local transport systems. This means, the nearest pharmacy is the one which can be reached in the shortest time. This problem amounts to a route planning problem. The system must compute the shortest route from the location X to the pharmacies and choose the one with the shortest route. The route planner must take into account the transport networks (road maps, tram lines, bus lines etc.), as well as the context information about my current situation. ■

In fact, it turns out that the formalization of many other locational notions involve the solution of a route planning problem.

Example 1.3 Consider the query "give me all cities *between* Munich and Frankfurt". What does *between* mean here? If we take a map of Germany and draw a straight line from Munich to Frankfurt, it does not cross many cities. A more elaborate (and still too simple) formalization of *between* could be: in order to check whether a city B is between the cities A and C, compute the shortest route R_1 from A to B, the shortest route R_2 from B to C and the shortest route R_3 directly from A to C. If the extra distance $d = length(R_1) + length(R_2) - length(R_3)$, I need to travel from A to C via B, compared to the direct route from A to C, is small enough, B can be considered to be *between* A and B. Since the condition "is small enough" is not very precise, one could use the distance d directly to order the answers to the query. ■

Example 1.4 Consider a database with, say, all cinemas in Munich. A query could be "give me all cinemas in the south of Munich". The "south of Munich", however, has no precise boundaries. Any artificial boundaries may yield strange results for many users. A more natural encoding of "in the south of" is therefore a 2-dimensional fuzzy distribution over the Munich region. The fuzzy distribution could even be non-zero for places outside, but close to the Munich border. This fuzzy distribution gives the location of each cinema a fuzzy value, and the fuzzy values can in turn be used to order the answers to the query. ■

Example 1.5 Suppose a company looks for a building site for a new factory. The site should be *close to* the motorway. "Close to" does in this case of course not mean the geographic distance to the motorway. It means the time it takes for a car or for a lorry to get to then next junction of the motorway. The length of the shortest path to the next junction can be turned into a fuzzy value which, in turn can be used to order the answers to the query.

We turned the relation $close\_to(point, line)$ into a shortest path problem whose result is then turned into a fuzzy value as the result of $close\_to(point, line)$. ■

Example 1.6 Suppose the database contains a road map, together with dynamic information about, say, traffic jams. The information about traffic jams is usually not very precise. It could be something like "there is a traffic jam on the M25 of 2 miles length between junction 8 and junction 10".

If the M25 is taken as a straight line then the traffic jam is a one-dimensional interval whose location is not exactly determined. Instead, we have some constraints: length = 2 miles, start after coordinate of junction 8, and end before coordinate of junction 10.

Queries like "is there a traffic jam on the western part of the M25" gives then rise to a constraint solving problem. ■

1.2 Existing Approaches

In a number of research areas methods have been developed which can help solving "locational reasoning" problems. Since "locational reasoning" is an extremely broad notion, which has connections to many areas in computer science, we can only mention here a few ones.

On the very concrete side there are the Geographic Information Systems (GIS), i.e. databases and algorithms which deal with concrete geographical data, road maps, land coverage etc.

'Shortest path in a graph' algorithms have been developed to solve the path planning problems, for example in transportation networks. The path planning problem in a concrete 2- or 3-D environment is one of the robot navigation problems, and there are a number of more or less practically useful algorithms to solve it [5].

The 'shortest path in a graph' algorithms do not take into account context information about the traveler. It is a totally different situation if the traveler has a car available, or if he depends on public transport systems. One way to use context information in a shortest path algorithm is to construct a problem specific graph for the shortest path algorithm. For example, if the traveler has a bicycle, the system might construct a graph consisting of paths and roads, together with those railway and bus lines where a bicycle can be taken into the coaches.

The GIS techniques depend on the availability of concrete coordinates. If coordinates are not available, symbolic data representation and reasoning is necessary. One of the symbolic locational reasoning systems is the 'region connection calculus' (RCC8, [4]). It conveys the ideas of Allen's interval calculus from the one-dimensional case to the two-dimensional case. RCC8 provides basic relations between two-dimensional areas and has rules for reasoning with the relations.

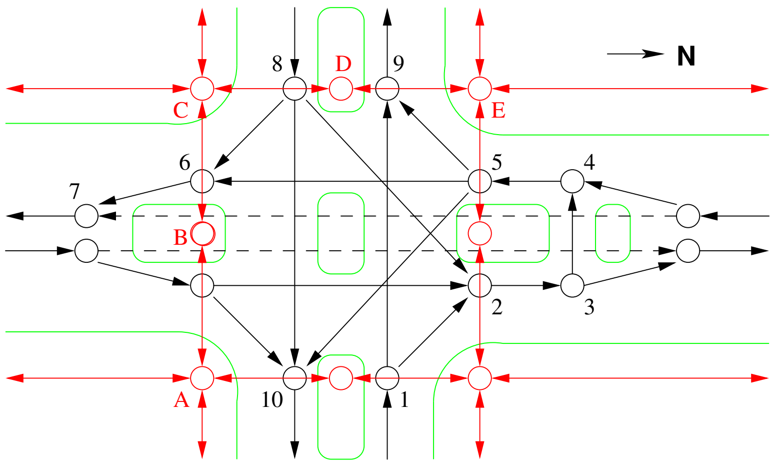
A very general knowledge representation and reasoning technique are the Description Logics [3], with OWL ass its WWW version [2]. In Description Logics one can define 'concepts', which correspond to sets of objects, and one can relate individuals to the concepts. The formula (1) is an example of a concept definition in a Description Logic.

1.3 Graphs, Graph Transformations and Ontologies

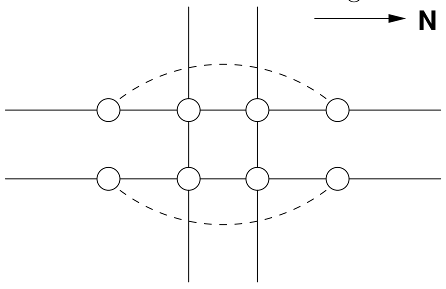
The examples in the introduction show that "locational reasoning" is a very heterogeneous subject. Therefore we tried to develop a unified view of the area, which allows one to combine the various techniques and results in a single system. The basis of the unified view is the observation that in most of the approaches the data can be represented as graphs, and that there are close connections between the different types of graphs. We illustrate this observation with some examples.

Example 1.7 (Road Crossings) The first graph in Fig. 1 shows a detailed representation of an intersection of two streets, including an underpass (dashed lines) and pedestrian pathways (shown in red). This graph is suitable for guiding an autonomous vehicle through the area of the crossing. A simplified version of this crossing is shown in the second graph. It contains enough information for a standard navigation system.

Finally, one can collapse the whole road crossing into a single node of the road network. This is sufficient for path planning on a larger scale.



Detailed Road Crossing



Schematic Road Crossing

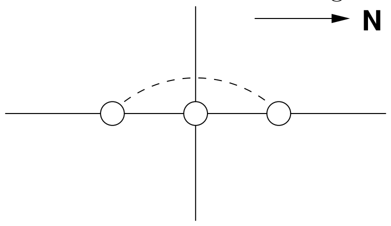
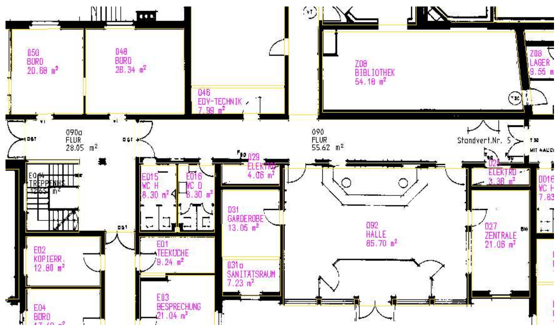


Figure 1: Road Crossing: Different Levels of Detail

In all three pictures we see the same road crossing, but on different level of detail. We are working at a language for describing how to generate the graphs with less detail from the graphs with more detail. ■

Example 1.8 (Floor Plans) Indoor navigation of autonomous vehicles requires a detailed floor plan, as shown in figure (1) of Fig. 2. In order to plan a way from, say, the entrance of the building to a particular office, such a detailed floor plan is not necessary. A simplified net plan, such as shown in picture (2) of Fig. 2 is much more suitable for this purpose. The simplified plan can be generated from the detailed floor plan.

(1)



(2)

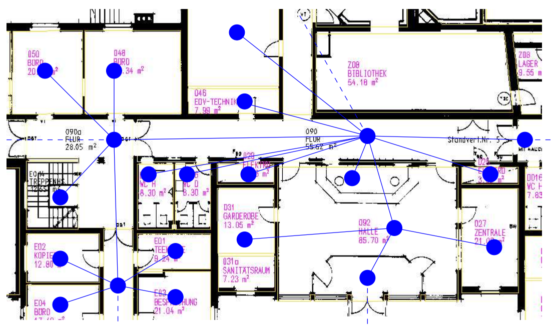


Figure 2: Plain Floor Plan without and with Network Overlay

Finally, one can collapse the whole building to a single node in a bigger city map. The node is sufficient for planning a path through the city to this building. ■

Example 1.9 (Symbolic Data Representation) This example shows the transition from GIS style data representation to a pure symbolic knowledge representation.

Figure 3 shows on the left hand side the boundaries of two of the German states, and some cities. The boundaries can be represented as polygons, and these are again just graphs. In the right picture the polygons are collapsed into single nodes of a graph. The relation 'polygon A is contained in polygon B' is turned into an NTTP edge (Non Tangengent Proper Part) of the new graph. The relation 'polygon A touches polygon B' is turned into an EC edge (Externally Connected) of the new graph.

■

The examples illustrate a number of observations

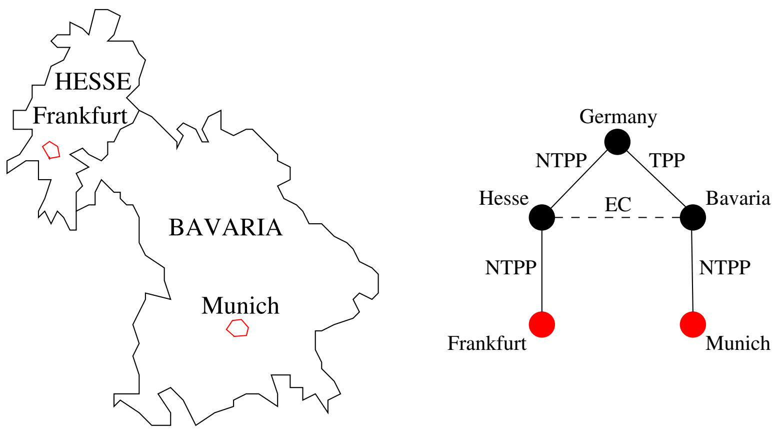


Figure 3: Symbolic Data Representation

1. There is a hierarchy of graphs. At the lowest level there are graphs with the concrete geographical details which are necessary for, say, guiding autonomous vehicles. At the highest level there are graphs which represent logical relations between entities.

2. There are correlations between the nodes and edges of the graphs at different levels of the hierarchy. These need not be a one to one correspondence. Usually a whole subgraph of a lower level graph corresponds to a single node or edge of the higher level graph. A typical example is the representation of the city of Munich in Example 1.9, as a polygon in the left hand graph and as a single node in the right hand graph.

3. A transition from a lower level graph to a higher level graph can be facilitated by identifying specific structures in the lower level graph, and transforming them into structures of the higher level graph with the same meaning. In example 1.7 this structure is a road crossing. In example 1.8 these structures are floors, doors, rooms etc. In example 1.9 these are cities, states etc.

   These structures are in general part of an *ontology*. In parallel with the development of the graphs, we therefore need to develop the corresponding ontologies. The elements of the ontology are the anchor points for controlling the graph transformations and for choosing suitable graphs to solve a given problem.

4. It is in general not a good idea to put all information into one single graph, even if it is information of the same level of detail. In a typical city we have, for example, a road map as a graph, the bus lines as a graph, the underground lines as a graph etc. We therefore need to consider collections of graphs with transition links between the graphs. Typical transition links between a road map and an underground map are the underground stations. The transition links, can, however, be little graphs themselves, for example the network of corridors and stairs in a big underground station.

5. The graphs at the higher levels of the hierarchy can and should usually be extended with additional information which is not represented in the lower level graphs. For example,

the graph in example 1.9 with the symbolic information about cities and states can be easily be extend by adding further cities and states.

1.4 A Road Map for the Development of Hierarchical Graphs

One of the most important results in this project will be the development of a technology of 'geospatial' knowledge representation with hierarchies of graphs. The hierarchy connects the coordinate based GIS like information processing with the logic based symbolic reasoning. The following steps are necessary to achieve this goal.

Step 1: Unified Representation of Graphs.
The structures at the different levels of the hierarchy are all graphs. Therefore there should be a unified representation of these graphs. The graphs need, however, be represented in different forms.

- We need a persistent representation of graphs which can be stored in files or databases.
- We need an in-memory representation of the graphs with a well defined application programming interface, probably similar to the DOM structures of XML data.
- We also need geometric representations of the graphs which can be used to display the graphs on the screen. As long as the nodes of the graph have coordinates, this is not a big problem. Graphs at the symbolic level of the hierarchy usually don't have coordinates. Fortunately there are well developed graph layout algorithms which we can use here.

Since graphs at different levels of the hierarchy can represent the same objects, road crossings, for example, it is very important to maintain the links between the same objects in the different graphs. These links enable algorithms to choose the level of detail they need for doing their computations.

It must also be possible to use the transition links between different graphs of the same level to join several graphs into one graph. For example, a route planner for somebody without a car may need a combined graph of all public transport systems.

As mentioned above, it should be possible to add extra information to the graphs, which is not derivable from graphs at the lower levels. In order to do this, we need to develop an *editor* for the graphs.

Step 2: 'Geospatial' Ontology.
We need to develop an ontology of interesting structures which can occur in the graphs (road crossings, roundabouts, floors, train stations etc.). The ontology is the anchor point for various auxiliary structures and algorithms, in particular:

- patterns which allow one to identify the structure in a graph, a roundabout, for example;
- transformation algorithms which simplify the structures to generate the nodes and edges in the graphs at the higher levels of the hierarchy;
- transformation algorithms which generate a graphical representation of the structures on the screen.

The ontology will also be used to annotate the structures in the graphs.

Step 3: Ontology of Graph Types.
The graphs at the different levels of the hierarchy provide the data for solving different kinds of problems. We need to classify the graph types, such that it is possible to choose the right graph for a given problem.

Step 4: Ontology of Means of Transportation.
A graph for a railway network, for example, represents only routes, but not the characteristics of the trains which are used on these routes. It can, for example, be important to know, which trains can take a bicycle on board, or which trains have wireless LAN on board etc. Therefore we need to develop an ontology for the objects which are connected with the graphs. If the graphs represent transportaion networks, this must be an ontology of the vehicles used on the network. If, on the other hand, the graph represents, for example, a local area computer network, it must be an ontology of the characteristics of the cables together with an ontology of the devices connected to the cables.

Step 5: Context Modelling.
In the introductory examples we showed that queries which require 'locational reasoning' need to take into account the context of the user. We must therefore develop a formal model of the context. The context can, for example, be the current situation of a human user: whether he has a car or not, whether he has luggage or not, his age and sex, and many other factors.

Step 6: Customized Graph Construction.
As we have seen in the introduction, many 'locational reasoning' problems require the solution of shortest path problems in a graph. The concrete graph which is relevant for the given problem, may, however, not be one of the graphs which are permanently available. It may be a combination of subgraphs from different graphs, and the combination may be determined by the context of the problem. Therefore we need to develop mechanisms for determining and constructing for a given problem the right combination of subgraphs as the input to the relevant problem solving algorithm.

Step 7: The Main Problem Solvers.
Finally we need to adapt or develop the algorithms for solving the main problems. These range from 'shortest path in a graph' algorithms until logical calculi for reasoning with symbolic information. Fortunately most of these algorithms are well developed and can, hopefully, be taken off the shelf.

2 MPLL – Multi Paradigm Location Language

Locational notions are so diverse that it is impossible to hard-code even the most important ones in a knowledge representation system. The alternative is therefore to develop a specification language for locational notions. The language can then be used to define application specific locational notions in a symbolic way. The specifications can, however, be compiled into executable code. The specification language must be expressive enough to define locational notions in an easy and intuitive way, and it must have the relevant data structures and algorithms built in.

Our – yet to be developed – language MPLL uses the time independent parts of the geotemporal specification language GeTS [9] as a kernel. MPLL extends the kernel of GeTS with location specific concepts. The built-in data structures of GeTS are various number types, one dimensional fuzzy intervals over the real numbers, labelled partitionings for modelling periodic temporal notions, calendar systems and durations. The number types and the one dimensional fuzzy intervals are also relevant for MPLL. All other data types are not needed. GeTS is a typed functional language with the usual control constructs, local variable bindings, but also assignments and a few other imperative constructs. These are also part of MPLL.

2.1 New Data Structures in MPLL

The first extension of the kernel of GeTS in MPLL are two-dimensional coordinates as built-in data structures. It is not a big problem to provide the various geographic coordinate systems with corresponding transformation functions.

The most important new data structure, however, are the hierarchies of graphs which we introduced on the last section. Quite a number of operations will be part of MPLL, from purely navigational operations to shortest path algorithms. An example for a specification in MPLL which uses these built-ins is the definition of *between*:

$$between(A,B,C) = \frac{length(path(A,C))}{length(path(A,B)) + length(path(B,C))} \tag{2}$$

$path(A,B)$ computes the shortest path from A to B. $length(path(A,B))$ yields the length of this path. Thus, $between(A,B,C)$ yields a value between 0 and 1. The result is 1 if B is exactly on the path between A and C. The longer the detour through B is, the smaller is the value of $between(A,B,C)$.

(2) is not really a realistic MPLL definition of *between.* First of all, it gives values close to 1 if B is close to A or C, but on the wrong side of A or C respectively. It is not difficult to refine (2) to fix this.

The second point is that the variables A, B and C in (2) need to be typed, for example with $between(place\ A, place\ B, place\ C)$. Here we seen another difference to GeTS. GeTS has only a fixed number of built-in types. In contrast to this, MPLL needs access to an ontology, such that the concepts of the ontology can be used as types in the language. ‘place would be such a concept, with subconcepts, for example, ‘city’, ‘town’ , ‘village’ etc. The MPLL compiler and the MPLL runtime engine will therefore have access to an ontology server such that the concepts of the ontology can be imported into MPLL at compile- and run time. It is obvious that the ontology which is imported as MPLL types is also used as attributes in the graphs, such that, for example, a variable *place A* can be bound to a node in a corresponding graph which is labelled, for example, with *type city*.

2.2 Contexts in MPLL

The function $path(A,B)$ in (2) is in fact under-specified. A shortest path in a transport network between A and B depends on the situation of the traveler. It makes a big difference, if, for example, he uses a car, or depends on public transport. There are two ways to deal with this. Either MPLL provides primitives for constructing traveler specific graphs, and calls $path(A,B,G)$ where G is the customized graph. Alternatively, MPLL passes context information about the traveler to the *path* function: $path(A,B,context)$. In both cases we need a

‘Context’ data structure in the language. The context can be used to model a traveler, but this is only a special case. In order to be as general as possible, context information will be represented as sets of key-value pairs $k = v$ where v can be a concept of an ontology. Examples are: $traveller = person \wedge gender = male \wedge age = 30 \wedge \ldots$, $means\_of\_transportation = BMW \wedge \ldots$. At this point MPLL needs access to an ontology server with ontologies about persons, vehicles etc.

3 Uncertainty

Many locational expressions are imprecise. Mathematical models of imprecision distinguish various kinds of impreciseness. Consider the phrase ‘there is a traffic jam of 2 miles length at the M25 between junction 10 and junction 12. It describes a real situation which is completely determined. However, our knowledge about the situation lacks some information, namely where exactly the traffic jam is. Mathematical formalizations of such situations use constraints and constraint reasoning.

The phrase ‘the accident happened at the M25 close to junction 10’ also describes a real situation which is completely determined. The expression ‘*close to*’, however, is so imprecise that constraint reasoning is not applicable. Alternatively one can use fuzzy distributions to map the distance to junction 10 to fuzzy values, i.e. real numbers between 0 and 1. The fuzzy value for distance x can be interpreted as a kind of probability that the accident happened at distance x from junction 10.

A combination of constraint reasoning and fuzzy distributions is necessary for modelling the phrase ‘there is a traffic jam of about 2 miles length at the M25 between junction 10 and junction 12’. ‘*about 2 miles length* expresses that the exact length is unknown. This can also be modelled with a fuzzy distribution for the length. The fuzzy values have their maximum 1 for exactly two miles length, and decrease for values greater than 2 miles and smaller than 2 miles. Since the exact location of the traffic jam is not known, but constrained between junction 10 and junction 12, constraint reasoning is necessary for this aspect. The combination of fuzziness for distances between points and constraints for the exact location of points yield so called fuzzy constraint networks. There are algorithms for solving fuzzy constraint networks. At the current state of the project, we have not investigated whether and how they can be used for our purposes.

Uncertainties have also been modelled with various other mathematical and logical formalisms (probability theory, Dempster-Shafer theory, plausibility theory, default reasoning, epistemic logics etc.). It goes beyond the scope of this report to investigate all these approaches in the context of locational reasoning. Here we concentrate only on one method: fuzzy sets.

3.1 Fuzzy Sets

The prototypical fuzzy locational expression is ‘*close to*’.

‘*Close to* junction 10 at the M25’ indicates a one-dimensional fuzzy distribution along a line (the M25).

‘*Close to the North Pole*’ indicates a two-dimensional fuzzy distribution around a point.

‘*Close to the motorway M25*’ indicates a two-dimensional fuzzy distribution around a line (the M25).

‘*Close to London*’ indicates a two-dimensional fuzzy distribution around an area (London).

In principle these four cases are very similar. The distance from a given point x to a given point/line/area is turned into the fuzzy value associated with x. The distance may be measured either with geographical coordinates, or with the metric which is determined by the local transport systems. Turning distances into fuzzy values between 0 and 1 has some advantages. First of all, the function which maps distances to fuzzy values introduces an assessment of the distance. If it is, for example, an exponentially decreasing Gaussian distribution (cf. Fig. 3.1), it makes points close to the point/line/area equally important (fuzzy value close to 1), and points further away equally unimportant (fuzzy value close to 0). Secondly, the fuzzy values are a kind of normalization which can make things comparable which are usually not so easily to compare. For example, the two phrases 'close to junction 10 on the motorway M25' and 'close to my office building in the city' refer to distances which are very different in the two cases. If, however, both are mapped to the same fuzzy value, say 0.9, this means really pretty 'close to'.

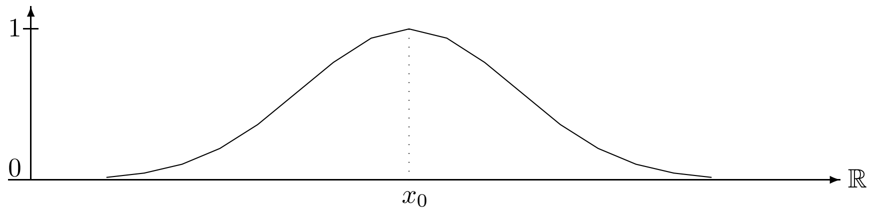


Figure 4: Gaussian Shape

An n-dimensional fuzzy distribution is an $n+1$-dimensional structure, where the $n+1$st dimension is the fuzzy value. Fig. 5 shows the three-dimensional structure which represents a two-dimensional fuzzy distribution around a line segment.

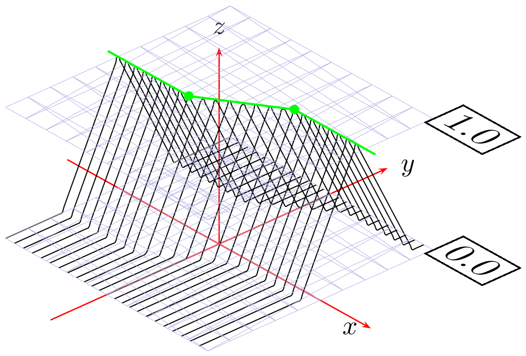


Figure 5: Linear fuzzification of a line segment

The one- and two-dimensional fuzzy distributions are of particular importance for locational reasoning.

3.2 One-Dimensional Fuzzy Distributions

A phrase like '*close to* junction 10 at the M25' contains the fuzzy notion 'close to'. A line in a two-dimensional plane is essentially a one-dimensional structure. The two-dimensional coordinates of the line points can be turned into one-dimensional coordinates by taking the distance from a specific point S along the line as coordinate (cf. Fig. 6).

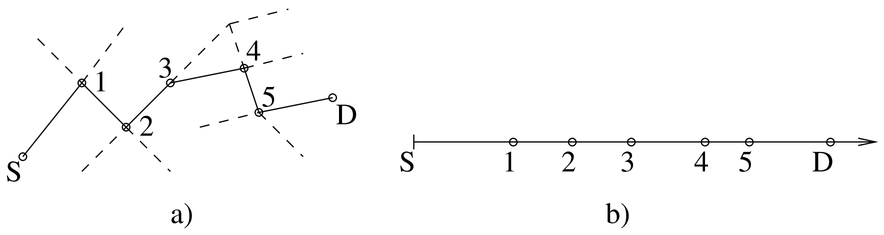


Figure 6: Representation of a Route as Straight Line Segment

If we associate each point of the line a fuzzy value, then the one-dimensional representation of the line offers the possibility to use the purely one-dimensional fuzzy interval mechanisms, in particular the FuTIRe library which has been developed for fuzzy temporal intervals [8].

The fuzzy encoding of 'close to' involves a mapping of distances to fuzzy values. This mapping is very subjective and context dependent. There are, however, one-dimensional fuzzy distributions which can be based on objective criteria. In a region with poor GSM coverage, for example, the fuzzy value along the route could indicate the reception quality for mobile phones, ranging from 0 (none) to 1 (excellent). Converting distances to GSM base stations along the route into fuzzy values provides a simple way to ascertain GSM reception for any point along the route. Many other route characteristics can be modelled this way: inclines, friction coefficients, curve radii, terrain attributes such as flora or housing density, and more.

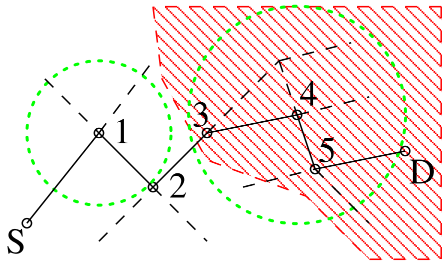
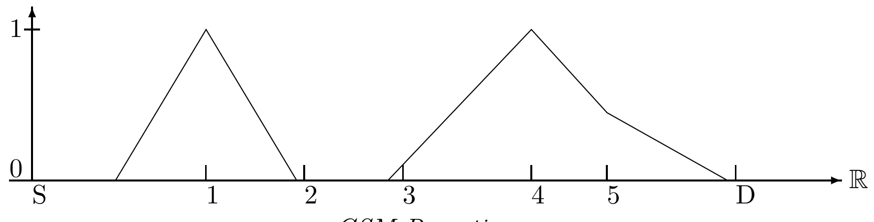


Figure 7: Extended Route Features

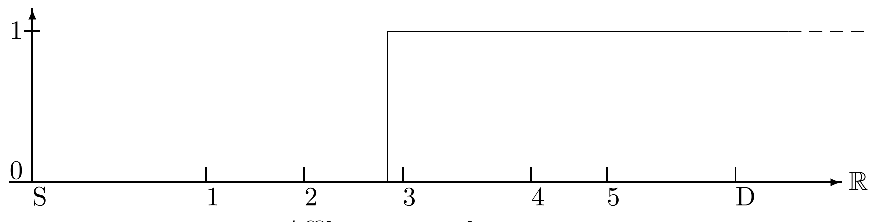
Figure 7 shows an enriched version of figure 6. The dotted green lines denote the maximum range of GMS base stations, an urban area is marked by red hatching. For example the quality of GSM signal reception for any location along the route can be now deduced by calculating the distances to the nearest GSM transmitter in the vicinity. The result is a fuzzy interval like the following[1]:

[1]Technically, reception quality and attenuation is not linear over distance and is of course depending on several more complicated factors, but for demonstration purposes this approximation will suffice.



GSM Reception

An interval marking urban areas can be applied in a similar way. The following diagram shows the urban boundaries as a crisp interval along the route from S to D:

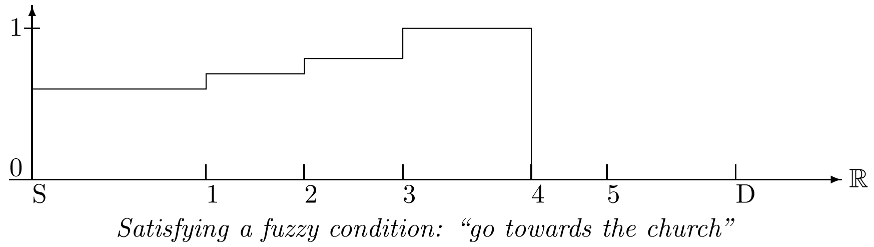


Affiliation to urban areas

3.2.1 Integration of Landmarks

Fuzzy intervals such as the above can also be used to integrate *landmarks* into a route. Landmarks are a "monument or material mark or fixed object used to designate a land boundary on the ground: any prominent object on land that may be used to determine a location or a direction in navigation or surveying" [1] and "point references considered external to the user" [7]. Landmarks serve primarily as navigation and routing aids. As they are only very rarely part of the route itself incorporating landmarks into route data structures poses an interesting problem.

Landmarks are used for a variety of purposes. They can denote a location where a certain action is to be performed: "Turn right at the supermarket." They also serve as directional aids: "Go into the direction of the cathedral.", "Follow the river on your right hand side." Fuzzy representations are sometimes particularly well suited to handle references to landmarks. Directions for example (see section **??** are rarely crisp. Therefore a fuzzy interval which denotes the fulfillment of a typical landmark notion such as "go uphill" or "go towards the church" could only be computed with difficulties by crisp means (slight deviations are problematic), while fuzzy representations are quite elegant. Assuming that on a smaller scale example of the scenario described in Fig. 7 the church is located at junction 4, the fuzzy intervals for the different route segments could look like this:



Satisfying a fuzzy condition: "go towards the church"

A computation by crisp methods needs to somehow include special treatment of "near misses". If the instruction is "go towards the church", would a segment which deviates from the absolute direction to the church by only a few degrees be disregarded? What happens if there are no other alternatives, in particular not a single one satisfying the instruction? What happens if there are several equally (i.e. perfectly well) suitable alternatives? These questions also arise when using fuzzy logic, but solving them is much easier.

3.3 Two-Dimensional Fuzzification

In the same way a one-dimensional structure can be fuzzified by transforming it into a two-dimensional structure, a fuzzified two-dimensional structure can be represented by a three-dimensional structure. Two-dimensional shapes represent many different elements commonly found in a map. A series of line segments can represent a highway, railway line or a river, but also abstract elements like border lines or flight connections. Polygons denote all elements which occupy an area of some sort, either real or abstract: cities, lakes, woods, farm land, districts, and so on.

Fuzzification of these shapes is especially important because of their importance for navigation and their common use in interpreting and communicating geospatial data. Real-life expressions like "near the river" or "in the south of Munich" can only be processed correctly, if there exists an equivalent counterpart in the model representation. The first expression means that a linear shape (the river) has to be expanded by fuzzification into a polygon which encompasses the area denoted by the term "near" in a fuzzy way. To be more exact, the two-dimensional shape becomes a three-dimensional shape which looks like a ridge (see Fig. 5). The second expression implicates the transformation of a polygon (the city of Munich, simplified in Fig. 8 a)) into a trapezoid-like shape. In two separate steps, the polygon has to be clipped (Fig. 8 b), c)) to satisfy the notion of "the south of..." and the resulting polygon must be fuzzified in order to represent the notion of "within" (Fig. 8 d)).

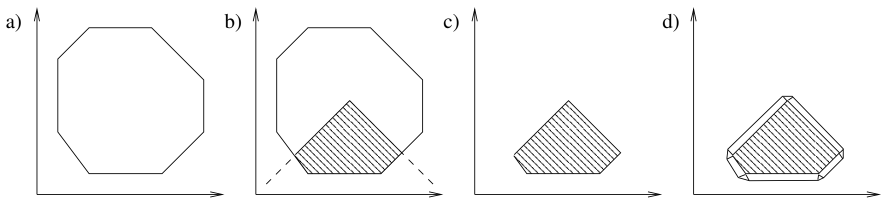


Figure 8: 'In the South of'

3.4 Representation of Fuzzy Distributions

Fuzzy distributions are determined by their membership functions. An important question for a concrete implementation is whether it is necessary to represent a fuzzy distribution as a concrete data structure, or whether it suffices to be able to apply the membership function to particular elements of the base set. In most cases this will be sufficient. For answering a query 'give me all cinemas in the south of Munich', for example, it is not necessary to store the three-dimensional representation of the south of Munich (Fig. 8 d). It is completely sufficient to apply the membership function to the coordinates of the cinemas.

A representation of whole fuzzy distributions may become useful if, for example, it is necessary to integrate over the fuzzy distribution. Consider, for example, the statement 'the traffic jam was between *around* junction 10 and *around* the road works at junction 12'. 'around junction 10' and 'around the road works at junction 12' can be modelled as one-dimensional fuzzy distributions as in Fig. 9. The fuzzy value for the traffic jam (car density, for example) is maximal at a location x which is definitely after 'around junction 10' and before 'around the road works at junction 12'. 'Definitely after around junction 10' means that the normalized integral over the fuzzy distribution 'around junction 10' must be 1. 'Definitely before around the road works at junction 12' means that the normalized backwards integral over 'around the road works at junction 12' must also be 1. The dashed line in Fig. 9 indicates the intersection of the two integrals as a representation of the traffic jam.

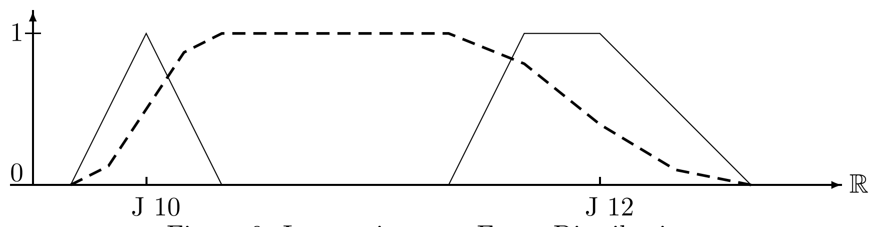


Figure 9: Integration over Fuzzy Distributions

The FuTIRe library has a data structure for representing one-dimensional fuzzy intervals as polygons together with various operations, in particular integration functions. We have not yet investigated whether similar data structures and operations are necessary and possible for the two-dimensional case.

Fuzzy distributions must of course be integrated with the hierarchies of graphs we introduced in Section 1.3. For the graphs whose nodes are labelled with concrete coordinates, this is only a technical problem which can be solved at the implementation level. It is not yet clear whether such an integration is necessary and possible for the abstract graphs which contain only symbolic knowledge.

4 Current State of Research on Locational Reasoning in WG A1

We started our research in this area with the study of existing approaches to 'locational reasoning', from GIS systems to very abstract logical calculi. The next step was the development of a common framework which allows one to combine the different levels of abstraction. We are now convinced that the graph hierarchy, and in particular the possibility to navigate through this hierarchy is the right approach.

Concrete locational notions, like 'close to' or 'in the south of' etc. cannot be hard coded into the implementation. Our approach is therefore the development of the MPLL specification language. The kernel of this language is identical with the kernel of the GeTS language for temporal notions. Therefore we have already an implementation to start with. Quite early it became clear that essential parts of the locational reasoning mechanism are ontologies. Therefore we started the development of an ontology for transport networks.

4.1 Transport Network Ontology (OTN)

An essential foundation for interoperable applications is a holistic concept of the underlying structures of the data to be processed, i.e. an ontology. The purpose of the Transport Network Ontology (OTN) is to provide such a foundation for applications which deal with locations, locational relationships and mostly with the aspects of locomotion and transportation.

4.2 The Origins of OTN

Different parties have worked on standards and interfaces in geographic data interchange since the late eighties. In 1993, the technical commission TC204 of the *International Organization for Standardization* (ISO) [?] began work on *Intelligent Transport Systems* (ITS) [?]. The aim of their working group 3 was to review existing regional standards, which revealed to be highly heterogene. While the *Japan Digital Road Map Association* (JDRMA) [?] mainly worked on standards catering for navigation systems and the necessary optimizations therefore, the American *Spatial Data Transfer Standard* (SDTS) [?] was designed to facilitate the description of records, but not the standardization of content. In Europe, the *Geographic Data File* (GDF) [?] was developed as an extensible and application-independent data model for transport systems. Subsequently, seven countries[2] continuously revised and extended GDF, which led to the release of GDF 4.0 on 21. March 2002 as the official ISO standard ISO-14825 [6] for geographic data interchange in transport applications.

While the underlying model of GDF mainly includes a thorough representation of car traffic and road networks, other modes of transportation have received less attention. Additional elements, such as services or public transport, are not included in GDF. OTN incorporates the comprehensive model of road networks underlying GDF, and extends the ontology to compensate for the neglected fields. Further extension is not only possible, but also desirable, since there cannot be a complete ex-ante model of all traffic and transport related affairs – nor for any other domain for that matter. To be usable with today's web infrastructures, OTN is specified in the *Web Ontology Language* (OWL). OTN contains some extensions which are not present in GDF: schedules, services and meteorology.

4.3 Schedules

OTN was developed as an integrated approach to modelling private and public transport. Therefore one of the most important features is the specification of schedules. In GDF the specification of schedules is limited to providing a time frame (start and end time) and a network segment (road or ferry segment) to convey that for example the ferry from Staten Island to Manhattan operates from 04:30 am to 11:30 pm. Further specification of travel times, intervals and such is not possible.

Because of the importance for (multi-modal) routing, OTN facilitates the definition of departure times, travel times and time frames. A typical segment or connection in public transport has an attribute *"timetable"*, which holds a series of schedules. Each schedule is valid during *"validity_Period"*, i.e. the time frame in which the respective means of transportation operates, while *"loop_Time"* defines the interval. *"starts_at"* contains the starting node (which can be located at either end of an edge) and *"travel_Time"* contains the regular or individual duration of travel. Optionally, a *"waiting_Time"* indicates an idle time before departure. A ferry, which

[2]Australia, Canada, Germany, Japan, Korea, the Netherlands and the U.S.

commutes hourly between 06:30 am to 06:30 pm from node `A` to node `B`, has a travel time of 30 minutes, and can be loaded 20 minutes prior to departure, could have a schedule like the following:

```
<Timetable rdf:ID="Timetable_A-B">
  <starts_at rdf:resource="#A"/>
  <waiting_Time>m20</waiting_Time>
  <loop_Time>h1</loop_Time>
  <travel_Time>m30</travel_Time>
  <validity_Period>
    <Validity_Period rdf:ID='validity_Timetable_A-B'>
      <time_duration>h12</time_duration>
      <starting_Date>h6m30</starting_Date>
    </Validity_Period>
  </validity_Period>
</Timetable>
```

4.4 Services

OTN caters for different aspects beyond those pertaining to routing and navigation. *Services* represent one of these aspects.

GDF generally introduces the notion of services, although – among a series of proposed services – only one is implemented: the service *"Entry_Point"* defines the access to a service.

OTN includes most of the GDF proposed services and provides further extensions. The attribute *"is_Accessible_at"* renders the GDF-service *"Entry_Point"* useless[3], therefore it has not been taken up in OTN. One, for the purpose of OTN very important, service is called *"Transfer_Service"*. It describes means to change the transport vehicles, for example, from car to train. Parking places, for example, are modelled as part of transfer services.

4.5 Meteorology

New in OTN is the possibility to store weather information. There is the topic *'Meteorology'*, which is subdivided into the classes *'Temperature'* and *'Weather'*. These are subclasses of *'Face'* and define an area with the actual temperature and the kind of weather: 'snow', 'sleet', 'hail', 'dew', 'rain', 'shiver' and 'storm'.

5 Summary

'Geospacial Reasoning' is an extremely broad area. In this deliverable we presented our approaches to cover enough aspects from this area such that it is useful for Semantic Web applications. An important building block is the hierarchy of graphs which combines very low level coordinate based computations with abstract symbolic reasoning. Another building block is the ontology of transport networks OTN. The graphs and the corresponding algorithms, together with OTN, provide the built-ins for the specification language MPLL. MPLL is a functional programming language, which, however, is mainly used to control the application of built-in

[3] *"Entry_Point"* only represents another service and it's accessibility.

algorithms, for example shortest path computations. With MPLL one can define customized locational notions, and evaluate them over the given data (maps, transport networks, user context etc.). The processing of uncertainty is certainly a major issue also for locational reasoning. So far we have investigated the use of fuzzy sets in this context. The FuTIRe library, which was developed for fuzzy temporal intervals can be used for one-dimensional fuzzy distributions in a higher dimensional space as well. It is easy to integrate the processing of two-dimensional fuzzy distributions which are only represented by their membership functions. This is sufficient as a first approximation to fuzzy reasoning in two dimensions.

References

[1] *EPA - U.S. Environmental Protection Agency: `http://www.epa.gov`.*

[2] OWL Web Ontology Language. http://www.w3.org/TR/owl-guide/.

[3] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications.* Cambridge University Press, 2003.

[4] Anthony G. Cohn, Brandon Bennett, John Gooday, and Nicholas Mark Gotts. Qualitative spatial representation and reasoning with the region connection calculus. *GeoInformatica*, 1(3):275–316, 1997.

[5] Kamal Gupta and Angel P. del Pobil, editors. *Practical Motion Planning in Robotics.* Wiley, 1998. ISBN: 0-471-98163-X.

[6] Iso/ts standard 14825: Intelligent transport systems – geographic data files (gdf) – overall data specification. `http://www.iso.org`, February 2004.

[7] Kevin Lynch. *The Image of the City.* MIT Press, June 15, 1960.

[8] Hans Jürgen Ohlbach. Geotemporal reasoning: Basic theory. Deliverable D1 of EU NoE Rewerse Working Group A1, 2004.

[9] Hans Jürgen Ohlbach. Implementation: Gets – a specification language for geo-temporal notions. Deliverable D10a of EU NoE Rewerse Working Group A1, 2005.