Towards Modeling Locations as Poly-Hierarchies

Hagen Höpfner and Maximilian Schirmer
Bauhaus-Universität Weimar
Media Department / Mobile Media Group
Bauhausstraße 11, 99423 Weimar, Germany
hoepfner@acm.org, maximilian.schirmer@uni-weimar.de

Keywords

Location, Location Modeling, Poly-Hierarchies, Enclave Problem, Multiple Belonging

ABSTRACT

Location models are formal descriptions of locations (i. e., named sets of geographical positions) as well as of semantic relationships among locations. There exist various location models that vary in the considered location information, their level of detail, the kind of modelled relations and the used formal representation. In fact, more complex location models cover more aspects required for implementing location-aware or location-dependent systems, but also require more complex algorithms. As each application domain requires only a limited set of features, limiting a model to those features helps to improve system performance. In this paper, we discuss a novel location model called location poly-hierarchies that models the belonging of one location to one ore more other locations. Furthermore, we present an approach for creating location poly-hierarchies from a given set of locations.

1. INTRODUCTION AND MOTIVATION

In February 2004, the authors of [5] stated: "The widespread deployment of sensing technologies will make location-aware applications part of every day life." Nowadays, location-awareness has become a key feature in a broad range of mobile information systems. Navigation systems, social network apps, tourist information systems, event information systems, shop finders and many more heavily rely on position data of their users. Consequently, almost all modern smartphones supply positioning techniques. However, a position determination, e.g, by the use of the Global Positioning System (GPS), enables a smartphone to calculate coordinates and accuracy, only. As, from the perspective of the user, a semantic location is more understandable than coordinate information, location-aware or location-based systems map position information to semantically meaningful locations. For example, the GPS coordinates 50.972 797 and 11.329 18 are precise but likely not meaningful for humans. They belong to the building which is placed in the Bauhausstraße 11 in Weimar, Germany and the assumption being that such information is usually more meaningful to the user. Hence, we should use a service that maps positions to locations. However, location-based applications differ in their required precision [10]. While the name of the city in which a user and/or a device is currently located might be appropriate for handling location-aware data processing in an event information system, a navigation system demands for exact coordinates, or street names at least.

The aforementioned example also illustrates another issue that is common for locations. In many cases, semantic locations form hierarchical structures. For example, in a hierarchical location model, earth is divided into continents, continents into countries, countries into states, states into cities, cities into streets and streets into street numbers, and so on. However, modelling locations as mono-hierarchies oversimplifies the reality and does not support multiple belongings. While, e. g., Weimar is part of Germany, Germany of Europe, etc., Istanbul is part of Turkey, but also of Europe and Asia. Please note, we focus on geographic belongs-to-relationships (subset and overlap) rather than on geopolitical ones. Location hierarchies benefit from the fact that determining low-level location information determines upper levels, too. Location poly-hierarchies cure the mentioned model incompleteness while keeping the benefit of a "fast" lookup of the correct path within the poly-hierarchy (cf., [11]). The research question addressed in this paper is:

> *How can one algorithmically create location poly-hierarchies from a given set of locations?*

The remainder of this paper is structured as follows: Section 2 surveys related work. Section 3 introduces the concept of location poly-hierarchies. Section 4 presents our approach for creating them. Section 5 discusses implementation aspects. Section 6 summarises the paper and gives an outlook on future research.

2. RELATED WORK

There has been a lot of active research on suitable models and representations for location data in the field of location models. Location models are a core component of location-based applications. They represent not only location information, but also spatial (or even spatio-temporal [7]) relationships in the data, help to express relative locations, proximity, and allow users to determine containment of locations or connectedness of relationships.

The authors of [13] present in great detail the broad variety of location models that has been developed in recent years of active research. A key factor for distinguishing and characterising location models is their way of representing spatial relationships. According to [1], they can be categorised into set-based, hierarchical, and graph-based models. Hybrid models that combine several aspects exist as well. Figure 1 presents an overview of the three main lo-

24^{th} GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 29.05.2012 - 01.06.2012, Lübbenau, Germany.

cation model concepts. In the illustrated examples, the set-based approach is the least expressive one, as it only models the fact that there are two distinct locations within a set of locations, and a set of coordinates is assigned to each location. The hierarchical model adds containment information, and the graph-based model adds connectedness and distance in the form of edge weights.

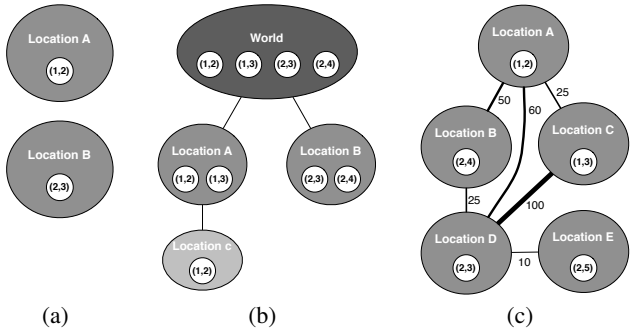

Figure 1: Examples for set-based (a), hierarchical (b), and graph-based (c) location models. The edge weights in the graph-based model represent distance information.

Hierarchical location models as a special case of set-based models represent containment relationships between different levels of the model and are widely used as basis for location-based applications [6, 2, 4]. Hierarchical models cannot represent distance information or directly encode proximity, but they have great advantages in traversal and for containment queries. They are also very close to the common human understanding of locations. As already stated in Section 1, the widely acknowledged segmentation of locations into administrative regions (country, state, city, street, and so on) is also a hierarchical model that heavily relies on containment information. On a city level, a lot of implicit information can be derived through the top-level relationship to a state or country (e.g., administrative language, local cuisine, prevalent religions).

3. LOCATION POLY-HIERARCHIES

The authors of [12] define the term "location" as follows: "Location of an object or a person is its geographical position on the earth with respect to a reference point." From our point of view, this definition is too restrictive, because geographic positions are points within a reference system. In contrast to a point, a location has a spatial extent. Another definition is given in [3]: "Geographic location is the text description of an area in a special confine on the earth's surface.". From a more theoretical point of view, an *area* is a set of geographical positions. So, we use a set-oriented definition: *A location is a named set of geographical positions on earth with respect to a reference point.*

For example, in a two-dimensional coordinate system[1], the location of a building, lets say a train station (ts), is given as set $L_{ts} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, where each position (point) $(x_i, y_i), 1 \leq i \leq n$ belongs to the train station building's area. We discuss the calculation of the point set in Section 5.1. Consequently, we can characterise the location of an object or a person as a relationship between sets. A person, lets say Tanja, is located at the train station if $L_{Tanja} \subset L_{ts}$ holds. Recent positioning technologies like GPS return only single points. However, we can interpret Tanja's current GPS coordinate as a set of positions of cardinality 1. As discussed in Section 1 it is a common understanding that locations have a hierarchic nature. The train station is located within a city, the city is located in a state, the state within a country, and so on. Using this subset-based location interpretation, $L_{Tanja} \subset L_{ts} \subset L_{Weimar} \subset L_{Thuringia} \subset L_{Germany} \subset L_{Europe} \subset L_{Earth}$ represents the location of Tanja as path in a location mono-hierarchy.

However, there exist various real-world issues that require a poly-hierarchical representation of locations. For example, Istanbul as the capital of Turkey, belongs to the continents Europe and Asia. Hence, $L_{Istanbul} \not\subset L_{Europe}$ and $L_{Istanbul} \not\subset L_{Asia}$ hold. The same issue holds for Russia, which is located in Europe and in Asia, too. Another problem results from enclaves. Kaliningrad, e.g., is part of Russia, but this information is insufficient to decide whether Kaliningrad is part of Europe or part of Asia. The solution for these problems is the use of set overlaps in combination with containment relationships that form a location poly-hierarchy.

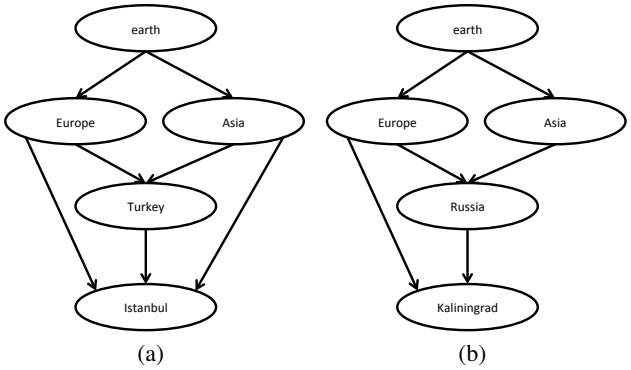

Figure 2: Examples for the Poly-hierarchical nature of locations: Istanbul (a), and Kaliningrad (b).

Figure 2 illustrates the simplified poly-hierarchies for Istanbul and Kaliningrad. From the definition of poly-hierarchies, we know that they can be represented by a directed acyclic graph $LPH = (V, E)$. Each node $v \in V$ is a location and each directed edge $e = (v_1, v_2)$ with $v_1, v_2 \in V$ represents that the child node v_2 belongs (semantically) to the parent node v_1. Each location poly-hierarchy has an unique root node $v_r \in V$ with $\neg\exists v_x \in V|(v_x, v_r)$, because the entire coordinate system is closed in case of locations (all considerable positions are elements of the set of all positions on earth). Finally, for each leaf node $v_l \in V$ that must not have any child node $\neg\exists v_x \in V|(v_l, v_x)$ holds.

From an implementation point of view, each node in the poly-hierarchy graph has the structure (n, P, C) where n is the name of the location L_n, P is a set of edges to the parent nodes and C is a set of edges to child nodes. The root node $r = (\text{earth}, P, C)$ is the only node in LPH for which $P = \emptyset \wedge C \neq \emptyset$ must hold. For leaf nodes $P \neq \emptyset \wedge C = \emptyset$, and for inner nodes $P \neq \emptyset \wedge C \neq \emptyset$ hold. For the concept described in the remainder of this paper, it is required to know the level of each node within this graph. The level $level(m)$ of a node m is defined as the number of nodes on the longest direct path from r to m, plus one. As illustrated in Figure 2(b), $level(Russia) = 2$ and $level(Kaliningrad) = 3$.

In a location mono-hierarchy each edge between a parent node and a child node represents the fact that all points of the location that corresponds to the child node are also elements of the location that corresponds to the parent node. However, as discussed before,

[1]For simplification purposes, we use two-dimensional coordinates in this paper. However, the formalism can easily be adapted to three dimensions.

it is not sufficient to use subset relationships only. The semantics of edges in the poly-hierarchical graph representation is as follows. Given a (child) node $c = (n_c, P_c, C_c)$ the following relations hold:

- For each parent node $p \in V$ referenced in P_c having *level*(p) with $\neg\exists p' \in P_c | p' \neq p \wedge$ *level*(p) = *level*(p') the edge $(p, c) \in E$ represents the subset relationship $L_{n_c} \subset L_{n_p}$.
- For each (parent) node $p \in V$ referenced in P_c having *level*(p) with $\exists p' \in P_c | p' \neq p \wedge$ *level*(p) = *level*(p') the edge $(p, c) \in E$ represents the overlapping relationship $L_{n_c} \cap L_{n_p} \neq \emptyset$.

In other words this means that per hierarchy level each edge between a single parent node and the child node means an subset relationship. If a child node has more than only one parent node in the same level, then those links represent an overlap relationship. Furthermore, we know that the in the latter case, due to the directed nature of the edges, the child node must be a subset of the union of the respective parent nodes (i. e., the conjunction of the overlap relationships per level holds).

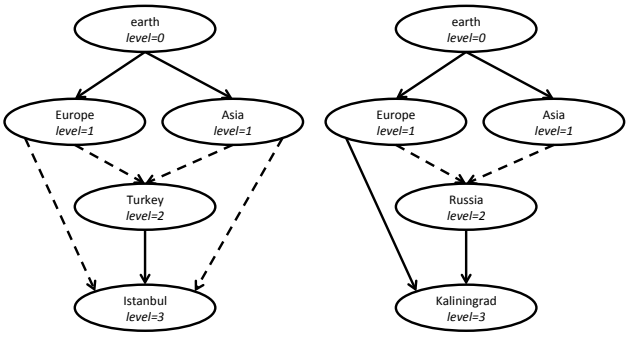

Figure 3: Examples for the semantics of edges in LPH, solid lines represent subset relationships and dashed lines overlaps.

Figure 3 illustrates the link semantics for the Istanbul and the Kaliningrad examples. As one can see in Subfigure 3(a), $L_{Istanbul} \subset L_{Turkey}$ as Turkey is the only parent node of Istanbul at level two. Since Istanbul has two parent nodes on level one (i. e., Europe and Asia), these links represent the overlaps $L_{Istanbul} \cap L_{Europe} \neq \emptyset$ and $L_{Istanbul} \cap L_{Asia} \neq \emptyset$. The facts that (in addition) $L_{Turkey} \cap L_{Europe} \neq \emptyset \wedge L_{Turkey} \cap L_{Asia} \neq \emptyset$ holds, also implies $L_{Istanbul} \subset L_{Europe} \cap L_{Asia}$. We could use this "transitive" conclusion in a similar way for the Kaliningrad example, too. However, as show in Subfigure 3(b) it would reduce the expressivity of this LPH. Kaliningrad is only part of Europe but not of Asia. Hence, the Europe node is the only parent node of the Kaliningrad node at level one.

4. LPH CREATION

As discussed in Section 3 one could create an LPH using overlap relationships only. We already pointed out that, in order to be as expressive as possible, an LPH must use as many subset relationships as possible. Algorithm 1 creates the LPH from a given set $\mathcal{L} = \{L_1, \ldots, L_k\}$ of locations. For illustration purposes, we assume that names of locations are unique. However, one could also use location IDs instead of the names to guarantee uniqueness.

Our algorithm uses four main steps. In the first step (lines 7-24), it creates two subgraphs, one $(V^{\subset}, E^{\subset})$ containing all subset re-

Algorithm 1 Creating a location poly-hierarchy from a location set

Input: $\mathcal{L} = \{L_1, \ldots, L_k\}$ // location set

Output: $LPH = (V, E)$ // location poly-hierarchy

```
01  def naive_lph_create(L):
02      V^⊂ = ∅ // init of the node set for subset relations
03      V^∩ = ∅ // init of the node set for overlap relations
04      E^⊂ = ∅ // init the edge set for subset relations
05      E^∩ = ∅ // init the edge set for overlap relations
06      L^V = ∅ // init set of already analysed locations
— STEP 1: FINDING OVERLAP AND SUBSET RELATIONSHIPS —
07      for each L_name1 ∈ L do
08          for each L_name2 ∈ L − {L_name1} − L^V do
09              if L_name2 ⊂ L_name1 then
10                  V^⊂ = V^⊂ ∪ {name1, name2}
11                  E^⊂ = E^⊂ ∪ {(name1, name2)}
12              fi
13              elif L_name1 ⊂ L_name2 then
14                  V^⊂ = V^⊂ ∪ {name1, name2}
15                  E^⊂ = E^⊂ ∪ {(name2, name1)}
16              fi
17              elif L_name2 ∩ L_name1 ≠ ∅ then
18                  V^∩ = V^∩ ∪ {name1, name2}
19                  E^∩ = E^∩ ∪ {(name2, name1)}
20                  E^∩ = E^∩ ∪ {(name1, name2)}
21              fi
22          done
23          L^V = L^V ∪ {L_name1}
24      done
25      if E^⊂ == ∅ then return(FALSE) fi // no root node found
— STEP 2: CLEANING UP E^∩ (REMOVING LOOPS) —
26      for each name1 ∈ V^∩ do
27          L^t = ∅; E^t = ∅ // temporary location and edge sets
28          for each e ∈ V^∩ do
29              if e == (name1, x) then
30                  L^t = L^t ∪ {L_x}; E^t = E^t ∪ {(name1, x)}
31              fi
32          done
33          if L_name1 ⊂ L^t then E^∩ = E^∩ − E^t fi
34      done
— STEP 3: CLEANING UP E^⊂ (TRANSITIVITY) —
35      for each name1 ∈ V^⊂ do
36          if ∃x, y ∈ V^⊂ ↔ {(x, name1), (y, x)} ∩ E^⊂ ≠ ∅ then
37              V^t = {z|z ≠ x ∧ (z, name1) ∈ V^⊂}
38              E^⊂ = E^⊂ − ⋃_{z∈V^t} {(z, name1)}
39          fi
40          V^t = {x|(x, name1) ∈ E^⊂} // all parents of name1
41          for each x ∈ V^t do
42              V^c = {y|(x, y) ∈ E^⊂ ∧ y ∈ V^t − {name1}}
43              if L_name1 ⊂ ⋃_{c∈V^c} L_c then
44                  E^⊂ = E^⊂ − {(x, name1)}
45              fi
46          done
47      done
— STEP 4: MERGING E^∩ WITH E^⊂ AND V^∩ WITH V^⊂ —
48      V = V^∩ ∪ V^⊂; E = E^∩ ∪ E^⊂
49      return(V, E)
```

lationships and one ($V^{\cap}, E^{\cap}$) for all (additional) overlap relationships. Figure 4 illustrates these subgraphs for the Istanbul and the Kaliningrad examples. Obviously, step 1 might generate redundant edges (cf., Figure 4(c)) `earth→Russia→Kaliningrad` and `earth→Kaliningrad`) or (later on) unnecessary edges (cf., Figure 4(a) `earth→Turkey`). Furthermore, it generates loops in the overlap subgraph as overlap relations have a symmetric nature (cf., Figure 4(b) and Figure 4(d)). We illustrated those needless edges using dotted arrows.

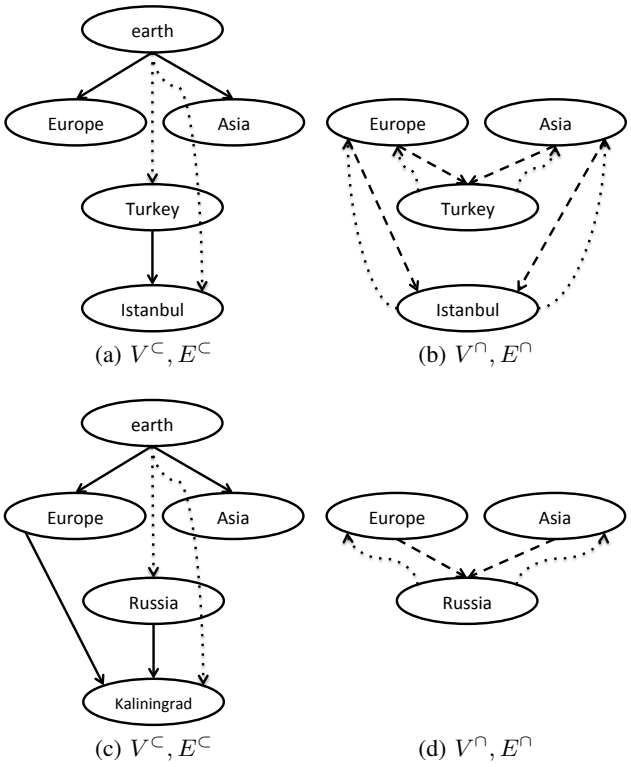

Figure 4: Intermediate graph(s) after step 1; (a,b) Istanbul, (c,d) Kaliningrad.

Step 2 (lines 26-34) of the algorithm cleans up the overlap subgraph, i. e., it breaks the loops. Therefore, each node of this subgraph is analysed (cf. Figure 5). If the location represented by a node is a subset of the union of all its direct child nodes, then we remove the outgoing edges.

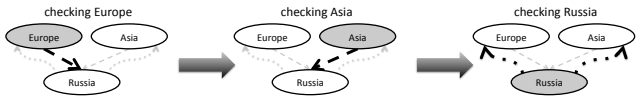

Figure 5: Step 2 for the Kaliningrad example; bold arrows highlight the edges analysed for currently handled node that is marked gray.

Figure 6 illustrates step 2 for the Istanbul example. After checking the nodes for Europe and Asia, the node for Turkey is checked. So far, it has the child nodes Europe and Asia. As all points of Turkey belong to Europe or Asia, both must not be represented by child nodes of the Turkey node. So, these edges are removed. The same procedure removes the edges `Istanbul→Europe` and `Istanbul→Asia`.

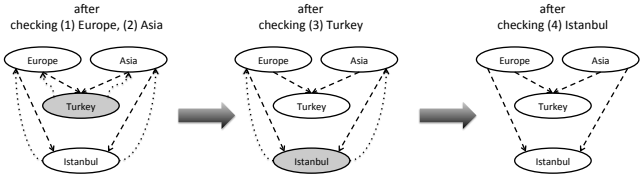

Figure 6: Step 2 for the Istanbul example.

Step 3 (lines 35-47) cleans up the subset subgraph using the transitivity property of subset relationships. In contrast to many graph optimisation approaches, we do not aim for reducing the number of edges in general, but for finding the minimal number of edges necessary for representing correct semantics (cf. Section 3). For the first optimisation (lines 36-39), we first calculate for each node the set of parent nodes having parent nodes themselves. Afterwards, we remove all direct edges from grandparents nodes as they can be reached through the parents. In our examples, this removes the edges `earth→Istanbul` and `earth→Kaliningrad`. A second optimisation (lines 40-46) checks whether a node is contained in the union of its sibling locations. If this is the case, we can remove the edge from the parent node (in our examples `earth→Turkey` and `earth→Russia`), even if this fragments the subgraph (cf. Figure 7).

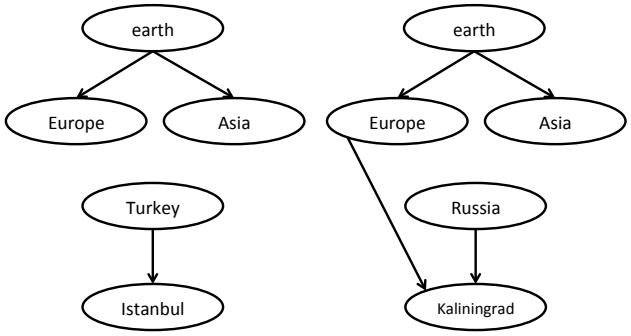

Figure 7: Intermediate graphs ($V^{\subset}, E^{\subset}$) after step 3.

Joining the subset subgraph(s) with the overlap subgraph, as it is done in the final step 4 (line 48), results in a connected LPH. The fragmentation, that might have happened in step 2, is cured as the removed edges resulted from an overlap relationship between the involved nodes. For our examples, Algorithm 1 therefore creates the location poly-hierarchies shown in Figure 3.

Limiting factors

Algorithm 1 requires a complete and closed set of locations. Without the location Asia, the Kaliningrad example graph would loose the LPH properties discussed in Section 3 as removing the Asia node also removes the edge `Asia→Russia`. Without this edge, the relationship `Europe→Russia` would be semantically wrongly interpreted as subset relationship. In fact, Algorithm 1 could be adapted in order to recognise this case through topologic sorting as the condition in line 33 would evaluate to false. Consequently,

$E^{\cap}$ would still contain loops after step 2. Furthermore, the algorithm does not support differently named locations with equal sets of position points (i. e., $L_a \subseteq L_b \wedge L_b \subseteq L_a$ holds). Step 1 would missinterpret this case as overlap relationship. Consequently, step 2 would nondeterministically remove one of the edges. A solution to this problem would be a cleanup phase that unifies those locations before starting Algorithm 1.

5. IMPLEMENTATION ASPECTS

As the "towards" in the title of this paper denotes, the presented results are subject to ongoing research. We were not able to finish the import of the evaluation database before the deadline. In fact, importing the OpenStreetMap database containing data about Europe (`http://download.geofabrik.de/osm`) using an slightly adapted version of the `osm2postgresql_05rc4.sh` script, which can be downloaded from `http://sourceforge.net/projects/osm2postgresql`, is still in progress. So far it took more than two weeks (PosgreSQL 9.1 [9] with PostGIS 1.5.1 [8] running on an iMac Intel Core 2 Duo 3.06 GHz, 4 GB 1067 MHz RAM, OS X 10.7.3). However, there are certain implementation issues that we find worthwhile to be discussed.

5.1 Point sets of locations

The formal definition of locations and location poly-hierarchies as discussed in Section 3 is based on set theory. From a theoretical point of view, those sets are infinite. A common approach to handle this infinity problem is to decrease the precision of set elements through quantisation. However, it would not be useful or even possible to physically store (materialise) all points of all locations. Hence, for the implementation of our approach we decided to use a polygon-based representation of locations, where a set of polygons describes the boundaries of the locations. In principal, all points that are inside of one of these polygons belong to the location. The OpenStreetMap data model provides an additional multipolygon relation (cf., `http://wiki.openstreetmap.org/wiki/Multipolygon_relation`) for representing more complex areas. It allows, e. g., for areas within a location that do not belong to this particular location. Hence, from an implementation point of view, a location is represented as a database view. The location name corresponds to the name of the view and the point set is defined by a database query resulting in the location outline (multi)polygon.

5.2 Set operations

Interpreting locations as (multi)polygons necessitates a different interpretation of the used set operations, too. As discussed in Section 4, Algorithm 1 has to check subset and set overlap relationships. Remember, a location L_a contains another location L_b if and only if $L_b \subset L_a$ holds. Hence, L_a must contain all points $(x_b, y_b) \in L_b$. For the polygon-based locations, the subset relation means that the (multi)polygon describing L_a must cover those of L_b completely. Similarly, an overlap relation of two locations implies that the boundary (multi)polygons overlap (and vice versa). Most geographical information system extensions, such as PostGIS, provide the corresponding operators (cf. [8]).

6. SUMMARY AND OUTLOOK

We presented a novel approach to model (geographical) relationships among locations. We extended the effective, but not complete location hierarchy approach in order to support enclaves and overlapping locations. We formally introduced the new location poly-hierarchy model and presented an algorithm for creating a location poly-hierarchy from a given (closed and complete) set of locations. We discussed limitations of the algorithm and also pointed out potential solutions to handle them. Finally, we discussed some issues that need to be considered for the implementation of our strategy. However, there are many open issues that lead to future research directions. First of all we plan to evaluate the algorithm with real world data. We expect that the algorithm works properly, but we are aware of the huge amount of data that needs to be processed. Hence, we will have to optimise the algorithm in order to avoid unnecessary (database) operations. Furthermore, we will research whether it would be better to explicitly keep the information about the type of the relationships. This extended graph model would, of course, ease the interpretability of the final location poly-hierarchy, but it would also increase the complexity of the model.

7. REFERENCES

[1] C. Becker and F. Dürr. On Location Models for Ubiquitous Computing. *Personal and Ubiquitous Computing*, 9(1):20–31, 2005.

[2] M. Beigl, T. Zimmer, and C. Decker. A Location Model for Communicating and Processing of Context. *Personal Ubiquitous Computing*, 6:341–357, Jan. 2002.

[3] Z. Dongqing, L. Zhiping, and Z. Xiguang. Location and its Semantics in Location-Based Services. *Geo-spatial Information Science*, 10(2):145–150, June 2007.

[4] F. Dürr and K. Rothermel. On a Location Model for Fine-Grained Geocast. In A. K. Dey, A. Schmidt, and J. F. McCarthy, editors, *UbiComp '03 Proceedings*, volume 2864 of *LNCS*, pages 18–35, Berlin / Heidelberg, 2003. Springer.

[5] M. Hazas, J. Scott, and J. Krumm. Location-Aware Computing Comes of Age. *IEEE Computer*, 37(2):95–97, Feb. 2004.

[6] C. Jiang and P. Steenkiste. A Hybrid Location Model with a Computable Location Identifier for Ubiquitous Computing. In G. Borriello and L. E. Holmquist, editors, *Ubicomp '02 Proceedings*, volume 2498 of *LNCS*, pages 246–263, London, UK, 2002. Springer.

[7] T. Kauppinen, R. Henriksson, R. Sinkkilä, R. Lindroos, J. Väätäinen, and E. Hyvönen. Ontology-based Disambiguation of Spatiotemporal Locations. In *IRSW '08 Proceedings*. CEUR-WS.org, 2008.

[8] OSGeo Project. *PostGIS 1.5.1 Manual*. `http://postgis.refractions.net/download/postgis-1.5.1.pdf`.

[9] The PostgreSQL Global Development Group. *PostgreSQL 9.1.3 Documentation*. `http://www.postgresql.org/docs/9.1/static/index.html`.

[10] B. N. Schilit, A. LaMarca, G. Borriello, W. G. Griswold, D. McDonald, E. Lazowska, A. Balachandran, J. Hong, and V. Iverson. Challenge: ubiquitous location-aware computing and the "place lab" initiative. In *WMASH '03 Proceedings*, pages 29–35, New York, NY, USA, 2003. ACM.

[11] M. Schirmer and H. Höpfner. Towards Using Location Poly-Hierarchies for Energy-Efficient Continuous Location Determination. In *GvD '12 Proceeding*. CEUR-WS.org, 2012. forthcoming.

[12] A. Y. Seydim, M. H. Dunham, and V. Kumar. Location dependent query processing. In *MobiDE '01 Proceedings*, pages 47–53, New York, NY, USA, 2001. ACM.

[13] J. Ye, L. Coyle, S. Dobson, and P. Nixon. A Unified Semantics Space Model. In *LoCA '07 Proceedings*, LNCS, pages 103–120, Berlin / Heidelberg, 2007. Springer.