Exploiting the Semantics of Location Granules in Location-Dependent Queries

Carlos Bobed, Sergio Ilarri, and Eduardo Mena

IIS Department
University of Zaragoza
50018 Zaragoza, Spain,
{cbobed,silarri,emena}@unizar.es

Abstract. The need for location-based services has motivated an important research effort in the efficient processing of location-dependent queries. Most of the existing approaches only deal with locations at maximum precision (e.g., GPS coordinates). However, due to imprecision or expressivity requirements, there are situations in which locations must be handled at different granularity levels (e.g., neighborhoods, cities, states, etc.). Indeed, whenever a set of locations are represented together as a granule, a meaning is implicitly given to the set. So, the use of different granularities brings different semantics to the location data.
In this paper, we propose the use of *semantic location granules* to enhance the expressivity of location-dependent queries. This is done by exploiting the semantic information that is asserted about different granularity levels. This information could be, for example, the cost incurred by a moving object to traverse a spatial area or a requirement to traverse a connection (e.g., need of a visa or passport). In particular, we propose: 1) an ontological model for describing the semantics inherent to location granules; 2) an upper-level ontology that can be extended and adapted to different scenarios; and 3) the use of a reasoner to exploit the semantics expressed in the ontologies, to make it possible to add new query constraints and so extend the expressivity of the queries.

1 Introduction

Nowadays, the interest in mobile computing has grown due to the ever-increasing use of mobile devices and their pervasiveness. The computing capabilities of mobile devices are also growing and users are demanding data access anywhere and at anytime. This has motivated, in the mobile computing field, an intensive research in *Location-Based Services* (LBS) [20]. These services provide value-added by considering the locations of the mobile users to offer customized information.

Processing location-dependent queries has been the subject of intense research [19], as it is a major building block of location-based services. Existing works on location-dependent query processing implicitly assume GPS locations for the objects in a scenario (e.g., [4, 10]). However, precise locations may be unavailable or even be inconvenient for the user. In those scenarios, it is useful to define the concept of *location granule* (similar to the concept of *place* in

[13]) as a set of physical locations. Some examples of location granules could be: freeways, buildings, offices in a building, etc.

The use of location granules to enhance the expressivity of location-dependent queries was first proposed in [17]. By using location granules in query constraints, the user is able to express queries and retrieve results according to the needed resolution. As described in that work, the use of location granules can have an impact on: 1) the presentation of results (location granules can be represented by using graphics, text, sounds, etc., depending on the requirements of the user), 2) the expressivity of the queries (the user expresses the queries according to his/her location terminology, and therefore the answers to those queries will depend on the interpretation of location granules), and 3) the performance of the query processing (the location tracking overload is alleviated when coarse location granules, instead of precise GPS locations, are used). In [16], the implications of using location granules in inside and nearest neighbour constraints were studied, and the granule-based query processing was also extended to deal with uncertainty, which led to consider *probabilistic granule-based inside queries* and *probabilistic granule-based nearest neighbour queries*.

However, thus far, the impact of location granules regarding location-dependent queries has only been studied from the point of view of query processing, leaving aside concerns about the semantics of location granules. Looking at the big picture, whenever a set of locations is represented as a location granule, we are assigning it an implicit meaning (e.g., cities, neighbourhoods, etc.). Even when the use of location granules is forced by the resolution given by the location service used, an implicit meaning arises (e.g., mobile phone cells). Making these meanings explicit would allow to further extend the expressivity of granule-based location-dependent queries. Besides, several properties and relationships could be considered once we have stated the exact interpretation of the location granules (e.g., when considering countries, different tolls could be established for traversing different boundaries).

In this paper, we propose to study the semantic dimension of location granules, extending their definition to obtain *semantic location granules*. This approach, which is complementary to our previous works on location granules [16, 17], allows to make their implicit semantics explicit and to take advantage of them to further extend the semantics of granule-based location-dependent queries. Firstly, we define the notion of *semantic location granule* and advocate the use of ontologies (which offer a formal, explicit specification of a shared conceptualization [11]) to represent them, by making explicit their properties, their relationships, and other logical rules that capture their semantics. Secondly, we analyze the most basic relationships that exist between location granules and propose a base ontology that can be extended and adapted to different scenarios. Thirdly, by using ontologies, and with the help of a reasoner [1], we propose an approach that allows extending dynamically the expressivity of the queries that can be processed. This extension of the expressivity can be done even at runtime by asserting new properties, relationships, and rules, into the granule ontology used. Using the inference capabilities of the reasoner, we make it pos-

sible to use this newly asserted knowledge to build new query constraints with the semantics explicitly stated by the user that posed the query.

The structure of the rest of the paper is as follows. In Section 2, we present two running examples that will accompany the explanations along the paper. In Section 3, the definition of *semantic location granules* is provided. In Section 4, we describe the base ontology that we propose. In Section 5, we focus on studying the impact of the added semantics in the processing of *inside* constraints. In Section 6, we present some related works. Finally, some conclusions and plans for future work appear in Section 7.

2 The Importance of Adding Semantics

In this section, we briefly overview the basics of the use of location granules in location-dependent queries, and then we show a couple of motivating examples.

2.1 Location-dependent Query Processing with Location Granules

For illustrative purposes, we use an SQL-like syntax along the paper to express the queries and constraints, which allows to emphasize the use of location granules and state the queries concisely. The structure of location-dependent queries is:

SELECT *projections*
FROM *sets-of-objects*
WHERE *boolean-conditions*

where *sets-of-objects* is a list of object classes that identify the kind of objects interesting for the query, *boolean-conditions* is a boolean expression that selects objects from those included in *sets-of-objects* by restricting their attribute values and/or demanding the satisfaction of certain *location-dependent constraints*, and *projections* is the list of attributes or location granules that must be retrieved from the selected objects. Specifications of the use of granules can appear in the SELECT and/or in the WHERE clause of a query, depending on whether those location granules must be used for the visualization of results or for the processing of constraints, respectively. If no location granules are specified, GPS locations are assumed.

In Figure 1, an example of how to process a granule-based inside constraint is shown. The general syntax of an *inside* constraint is *inside(r, obj, target)*, which retrieves the objects of a certain class *target* (such objects are called *target objects* and their class the *target class*) within a specific distance *r* (which is called the *relevant radius*) of a certain moving object *obj* (that is called the *reference object* of the constraint). Thus, for example, the constraint *inside(130 miles, gr(province, car38), gr(province, Car))* retrieves the cars in provinces within 130 miles of the province where car38 is located. To process that constraint: 1) the granule where the reference object (car38) is located is obtained and a buffering operation is performed to obtain the surrounding area within 130 miles; 2) the

granules that intersect that area are retrieved; and 3) the objects of the class *Car* that are inside those relevant granules are retrieved. For further details on how to process granule-based constrains we refer the reader to [16, 17].

(a) Step 1 (b) Step 2 (c) Step 3

Fig. 1. Inside with granules for the reference object and the target class: steps

2.2 Motivating Examples

In this section we present two examples that show that, although the semantics of the queries are extended by the use of granules, there is still a need for richer semantics. We will re-visit these examples in Section 5.3, once we have proposed a solution to represent the missing semantics.

Traffic Monitoring Example. The first example is a traffic monitoring application. In spite of using a traffic application as motivating example, we want to remark that our approach does not focus on query processing on road networks, as opposed to works such as [15, 22]. Thus, neither the trajectories of the moving objects in our approach are restricted to a fixed network nor the location granules are limited to represent roads. In fact, the granules we have modelled for the example can be considered as coarser representations of the motorways, which might include some fragments of surrounding roads. As we are not limited to work with road segments (arbitrary regions can be used to define the location granules), we can obtain also the objects that are likely to use the motorways without having to specify all the surrounding roads. Thus, we consider works that model objects moving on road networks complementary to our approach, as they could adopt our approach to add a separate layer to take into account semantic aspects that otherwise would be lost.

At a particular moment in time, let us suppose that we want an overview of the evolution of the traffic density that the A2 motorway is supporting, along with its surroundings, to be able to foresee possible traffic jams in that entry of Madrid. We have modelled the motorways entering Madrid as shown in Figure 2. A query with an inside constraint such as:

SELECT *COUNT(Car.id)*
FROM *Car*
WHERE *inside(50 miles,* $A2_1$*, Car)*

would seem perfect to obtain the number of cars going to Madrid using the A2 motorway; however, its semantics are not accurate enough because the buffering operation (see Section 2.1, and particularly the example in Figure 1) over the $A2_1$ granule could retrieve also cars that are going through the A1 and A3 motorways, which would introduce noise in the answer. Using granules for the target objects of the query:

SELECT *COUNT(Car.id)*
FROM *Car*
WHERE *inside(50 miles,* $A2_1$*, gr(Motorways, Car))*

does not resolve our problem as, despite retrieving the cars in $A2_1$ and $A2_2$, the system would also retrieve all the cars in $A1_1$, $A3_1$, $M50_1$, and $M50_4$.

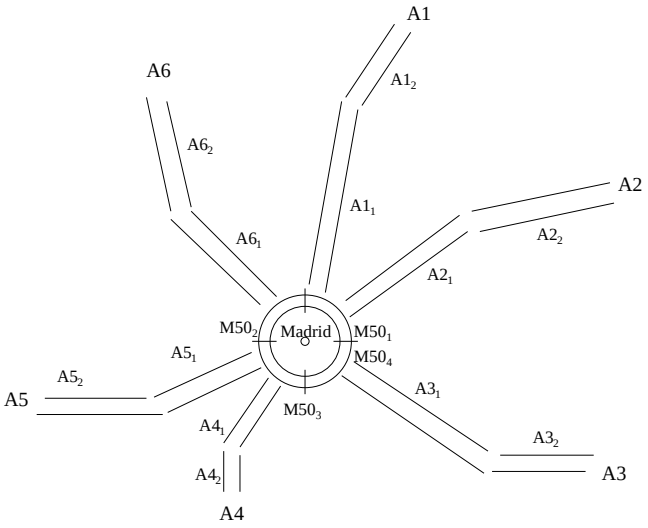

Fig. 2. Location granule mapping: motorways entering Madrid

Taxi Monitoring Example. Let us move to another example: a taxi cab monitoring application. Leaving aside the distance, the cost of taxi services usually depends on different aspects of the origin and target destination. Imagine a person that has already landed in the airport of Barajas (Madrid), and that s/he wants to call a taxi to go to the suburbs. S/he may also know that a taxi that might be closer in terms of physical distance could be more expensive than another that is farther, due to special taxes that are applied when taxis have to move into other zones. Using granules in the queries allows to consider the taxis according to the different zones, but not the additional costs that the taxi driver is going to charge to the user.

3 Semantic Location Granules

In previous works [16, 17], we proposed a definition of *location granule* from a physical point of view (see Definition 1).

Definition 1. *A* location granule *is a set of one or more geographic areas which identify a set of GPS locations under a common name.*

For example, the set of locations that belong to *Madrid* would be a location granule. In this section, we bring this definition to a higher abstraction level to capture the semantic dimension of location granules. Let us analyse what we do when forming a location granule: we group a set of locations and give them a name. This way we are also giving them a meaning (in this case, *Madrid* is a city and is the capital of Spain). The grouped locations become a new different entity as a whole, which could be related to other location granules in different ways besides spatial relations.

To formalize these notions, we advocate the use of ontologies [11]. Ontologies offer a formal, explicit specification of a shared conceptualization. Mathematically, an ontology can be seen as a tuple $< C, R, I, Rl >$, where C would be the set of concepts and their definitions, R the set of roles (relationships between concepts), I the set of instances that belong to the different concepts (the ontology population), and Rl the set of logical rules that are to be considered[1]. Regarding location granules and their properties:

- Focusing on the semantic dimension of location granules, the bare notion of location granule is itself a concept and the concrete location granules would be its instances. The set composed by the basic type of *granule* and the possible definitions of specialized types, along with the set of instances of those location granule definitions, map directly to the C and I sets, respectively. For example, *Madrid* would be an instance of the *cityGranule* concept.
- The different characteristics of the granules would be the attributes (*datatype properties*) of the concept. They might be or not geographical properties. For example, when considering cities, we could want to know their extension but also other characteristics such as possible fees that would be charged if we visit them (for example, in Mallorca, Spain, a special EcoTax fee is charged to travelers to fund environment protection). Moreover, the possible relationships between location granules would be the roles of an ontology (*object properties*). For example, considering countries, there could be special visa requirements to travel from one country to another. Both sets of attributes and relationships map directly to the R set.
- Finally, apart from the definitions of the granules, we can provide rules that allow to represent dynamic facts and extend the reasoning about the granules. This set of rules would directly map to the Rl set.

[1] In OWL [12], the reference language for ontologies in the Web, Concepts correspond to Classes, Roles correspond to ObjectProperties and DataTypeProperties, and Individuals to the homonym elements.

So, the use of ontologies fits perfectly the definition of location granule and its semantics. Making the semantics of location granules explicit turns them into *semantic location granules* (see Definition 2).

Definition 2. *A* semantic location granule *is a location granule with well-defined semantics, i.e., explicitly stated.*

The most important operator for a location granule is *inGr* (short for *inGranule*), which returns a boolean indicating whether a certain GPS location is within the granule.

Location granules will not be used in isolation in an application. Instead, they participate in groups of locations granules that provide an interpretation of the location space, conforming semantic layers or *semantic granule mappings* (see Definition 3).

Definition 3. *A* semantic granule mapping *is a set of semantic granules identified by a common name. It provides the global semantics of the location granules that participate in it.*

Following with the previous example, *Madrid* is a *cityGranule*, but it is assigned different global semantics if it participates in an application as being part of the *citiesOfSpain* or *capitalsOfTheWorld* sets of granules. Different semantic granule mappings could be defined over the same geographic area, and the user can choose the most appropriate for his/her context. The most important operators for location granule mappings are *getGrs* (*getGranules*) and *getGrsObj* (*getGranulesObject*), which return the subset of granules that contain a specified GPS location or object, respectively. In the rest of the paper, when referring to location granules and granule mappings, we mean semantic ones, and, for brevity, we will use *gr* instead of *getGrs*.

4 Modeling Location Granules with Ontologies

In the previous section, we have seen the definition of *semantic location granules*. In this section, we analyze the most basic properties of the granules and propose a base ontology for representing them and their associated granule mappings.

4.1 Base Ontology for Location Granules

This ontology (see Figure 3) is not meant to be complete, but a starting point to be extended and adapted to particular scenarios. The most basic properties that we identify are the following:

- *Contains*: it represents the physical inclusion of a granule inside another. For example, the granule *Spain* (the country) contains, among others, the granules *Madrid*, *Barcelona*, etc. (the cities). This property permits to establish a spatial hierarchy to organize the granule instances. *IsContained* is its inverse property.

– *Groups*: it represents the relationship that there exists between a granule mapping and the granules that make it up. For example, the granule mapping *Countries* would group the granules *Spain*, *France*, etc. *Participates* is its inverse property. Note that a granule can participate in several granule mappings.
– *Encapsulates*: it allows to establish hierarchies between granule mappings according to the granularity level. For example, the granule mapping *provincesOfSpain* could encapsulate *citiesOfSpain*. *IsEncapsulated* is its inverse property.

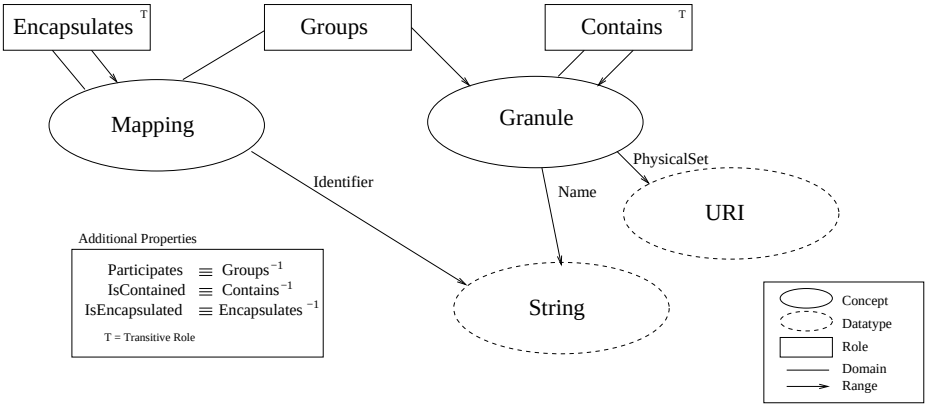

Fig. 3. Base ontology that can be dynamically extended

The rest of the properties are used to identify the granules (*Name*) and the granule mappings (*Identifier*), and to associate a granule to one or several sets of physical coordinates (*PhysicalSet*). The physical coordinates are also accessible at the semantic level to allow including statements about them in the asserted knowledge and, therefore, to allow the system to perform spatial reasoning.

Although this basic ontology may seem too simple, direct benefits can be obtained out of it. For example, even without any additional extension in the semantics, the presentation of results of the queries can be enhanced by exploiting the inclusion relationships to offer different views of the same answer set. Besides, we wanted to keep the model as simple as possible to make it easier to adapt it to the desired semantics.

4.2 Extending the Base Ontology

The proposed ontology can be loaded and handled by a Description Logics reasoner [1]. This allows us to make new knowledge available to the system dynamically by asserting it into the ontology. To extend the semantics we could

specialize the existing concepts or add new properties. To illustrate it, let us get back to the examples. In the taxi monitoring application, if we want to consider zones with no additional initial cost, we could assert the following:

$$
\begin{aligned}
&1) InitialCost : datatypeProperty \\
&\quad domain(Size) = Granule \\
&\quad range(Size) = float \\
&\quad functional \\
&2) InitialCostFreeZone := Granule\ and\ (InitialCost = 0)
\end{aligned}
$$

Let us focus now on the traffic monitoring example. We want to add a new role to the ontology which expresses the existence of a direct connection from one granule to another (an object can move from one granule to another directly without going through a different granule[2]). Of course, we also need to assert information about how the instances are related, but we can automate this process using propagation rules; so, in the first example, we would assert:

$$
\begin{aligned}
&1) DirectlyConnected : role \\
&\quad domain(DirectlyConnected) = Granule \\
&\quad range(DirectlyConnected) = Granule \\
&\quad symmetric, reflexive \\
&2) DirectlyConnected(A2_1, A2_2) \\
&\quad DirectlyConnected(A2_1, M50_1) \\
&\quad DirectlyConnected(A3_1, A3_2) \\
&\quad DirectlyConnected(A3_1, M50_4) \\
&\quad \ldots
\end{aligned}
$$

Along with the following Horn rule[3]:

$$
\begin{aligned}
&IsContained(?x, ?y) \wedge IsContained(?u, ?v) \wedge \\
&\wedge Participates(?y, ?z) \wedge Participates(?v, ?z) \wedge \\
&\qquad \wedge DirectlyConnected(?x, ?u) \longrightarrow \\
&\qquad\qquad DirectlyConnected(?y, ?v)
\end{aligned}
$$

The assertion of the rule allows to spread automatically (thanks to the use of a DL reasoner) the knowledge asserted, which makes it usable by other mappings that might be using the upper-level granules; for instance, a granule A2 which

[2] Note that this property is different from just sharing some edges in the boundary, as the existence of a shared boundary between granules in fact does not assure the possibility to traverse it.

[3] It can be expressed in SWRL [14], an extension of OWL to support rule-based inferences.

contains the A21 and A22 segments, would be automatically detected as directly connected to M50, as one of its inner granules is directly connected to one of the inner granules of M50. The inclusion of both *Participates* clauses in the Horn rule is to control that different granularities are not mixed.

5 Extending the Semantics of Inside Queries

In this section, we explain how to take advantage of the explicit semantics of the granules to enhance the expressivity of granule-based inside constraints. Firstly, we extend the operators defined for granules to be able to take into account the new semantics. Then, we analyze how this affects the definitions of granule-based inside constraints. Finally, the examples in Section 2 are reconsidered with the new available semantics.

5.1 Extending the GetGranules Operator with a Holds Function

Before extending the *gr* operator (getGranules, see Section 3), we have to introduce the *holds* function. *Holds* takes as input a semantic location granule (*slg*) and an expression that it has to satisfy in order to be considered relevant for the query. The allowed operators in the expression are *And*, *Or* and *Not*. The atoms used in the expression are the names of the concepts or the properties that we have previously added to the ontology, so *holds* has to be defined separately for evaluating concepts or properties between individuals or sets of individuals:

$$\begin{aligned} holds(slg, concept) = true &\Leftrightarrow slg \in concept \\ holds(slg, property, slg') = true &\Leftrightarrow property(slg, slg') \in property \\ holds(slg, property, \{slg_i\}) = true &\Leftrightarrow \exists slg' \in \{slg_i\} \ni holds(slg, property, slg') \end{aligned}$$

These definitions assume a Closed World scenario[4]. When dealing with ontologies and reasoners, we have to bear in mind that they usually assume Open World scenarios. This implies that the reasoner cannot infer anything which has not been directly asserted or inferred from the previously asserted axioms. This affects directly the definition and evaluation of the *holds* function. Assuming an Open World scenario, *holds*[5] would be defined as:

$$holds(slg, concept) \begin{cases} slg \notin concept \Rightarrow false \\ otherwise \Rightarrow true \end{cases}$$

$$holds(slg, property, slg') \begin{cases} property(slg, slg') \notin property \Rightarrow false \\ otherwise \Rightarrow true \end{cases}$$

$$holds(slg, property, \{slg_i\}) \begin{cases} \forall slg' \in \{slg_i\}, property(slg, slg') \notin property \Rightarrow false \\ otherwise \Rightarrow true \end{cases}$$

[4] If the Reasoner cannot retrieve the exact fact, it is evaluated to false.
[5] In an Open World scenario, whenever in doubt, *holds* is evaluated to true.

In this context, it seems to be more useful to have a Closed World instead of an Open World semantics as, in fact, we can control the facts to be asserted. So, once we have defined the holds function, and assuming Closed World, we redefine the *gr* operator as a three argument function with an optional fourth argument:

$$gr(m, obj, cond) = \{slg \mid participates(slg, m) \wedge inGranule(slg, obj.loc) \wedge \\ \wedge holds(slg, cond)\}$$
$$gr(m, obj, cond, \{slg_i\}) = \{slg \mid participates(slg, m) \wedge inGranule(slg, obj.loc) \wedge \\ \wedge holds(slg, cond, \{slg_i\})\}$$

being m an instance of granule mapping, *obj* an object, and *cond* a list of conditions that the granules comprising the answer set must fulfil.

Note that, although these semantics may seem very restrictive, the real potential resides in the expressive power that the concept definitions and the properties of the defined properties provide.

5.2 Extended Granule-based Inside Constraints

In this section, we focus on how the change in the *gr* operator affects the semantics of the three different situations in which it can be used in an inside query.

Inside constraint with a granule for the reference object. In this case, the corresponding *inside* constraint is now interpreted as follows:

$$inside(r, gr(map, obj, cond), target) = \{o_i \mid (o_i \in target) \wedge (\exists p \in GPS \mid \\ inGr(gr(map, obj, cond), p) \wedge distance(p, (o_i.loc.x, o_i.loc.y)) \leq r)\}$$

Why could we want to express a constraint over the granule where the reference object is? The reference object might be crossing a granule that does not hold some specific condition. For example, imagine a person walking in the rain in a zone in which taxis are not allowed to enter (or they do not want to because the zone is extremely dangerous). S/he only wants the taxi to get out from there, but it would make no sense to ask for taxis there as that granule is defined as a taxi-free zone; so, the application could automatically add this constraint and inform the user that is not going to be able to take a taxi. Moreover, granules might overlap each other, so it is possible for a location to be in two granules at the same time. In this way, we could restrict which granules are to be considered. Notice that the *inGr* operator (see Section 3) evaluates as *false* when the *gr* operator returns no granules.

Inside constraint with granules for the target objects. An *inside* constraint can include a granule mapping for the target class, in which case the constraint is interpreted as follows:

$$inside(r, obj, gr(map, target, cond)) = \{o_i \mid (o_i \in target) \wedge (\exists p \in GPS \mid inGr(gr(map, o_i, cond), p) \wedge distance(p, (obj.loc.x, obj.loc.y)) \leq r)\}$$

As mentioned previously, when no granule holds the condition, *gr* will return an empty list of granules and, consequently, *inGr* will be evaluated to *false* automatically.

Inside constraint with granules for the reference and target objects. This final situation is a mixture of the two previous cases. The new *inside* constraint is interpreted as follows:

$$\begin{gathered} inside(r, gr(map1, obj, cond1), gr(map2, target, cond2)) = \\ \{o_i \mid (o_i \in target) \wedge \ (\exists p_1, p_2 \in GPS \mid distance(p_1, p_2) \leq r \wedge \\ \wedge \ inGr(gr(map1, obj, cond1), p_1) \wedge \\ \wedge \ inGr(gr(map2, o_i, cond2, gr(map1, obj, cond1)), p_2))\} \end{gathered}$$

Note that we can specify different sets of conditions as well as different mappings for the reference object and the target class. This provides the user with huge flexibility to express the conditions over the granules involved in the query.

5.3 Reconsidering the Examples with Semantic Location Granules

With the newly added semantics, the user of the traffic monitoring application can now pose the exact query:

SELECT *COUNT(Car.id)*
FROM *Car*
WHERE *inside(50 miles,* $A2_1$*,*
gr(Motorways, Car, DirectlyConnected, $\{A2_1\}$*))*

Now, the system will only retrieve the cars that are in segments directly connected to the segment whose traffic we want to monitor.

On the other hand, the taxi user that wanted to save money could pose a query such as:

SELECT *Taxi.pos*
FROM *Taxi*
WHERE *inside(1 mile, me, gr(TaxiZones, Taxi, InitialCostFreeZone))*

to look for taxis within one mile in zones that would not imply initial additional costs. Notice that, to extend the expressivity of the queries, in the first example we are using a property and in the second one a concept definition in the constraints.

6 Related Work

Several works on spatial databases and geographic information systems deal with the problem of managing spatial data at different levels of detail (e.g., [5, 9]). These works focus on the problem of dealing with different levels of detail/specification of the spatial entities, and therefore they do not consider the use of locations at different granularities to enhance the expressiveness of queries. In the same field, the use of ontologies is mainly devoted to achieve data integration and interoperability between systems [2, 3, 8, 23].

In the field of pervasive computing, several works have emphasized the importance of adding semantics to location data [7, 21], although they do not look at this problem from the perspective of a user that wants to express queries using a suitable location granularity and with the required semantics:

- In [7], according to pervasive computing criteria, a taxonomy of types of locations is given. Therefore, locations are assigned implicit meta-semantics depending on the classification.
- In [21], the authors propose a location model that supports different expressive representations for spaces, spatial relationships, and positioning systems. However, this work focuses on positioning systems, with an emphasis in sensor data fusion. Thus, for example, the location measured by a sensor may have associated a symbolic representation. On the contrary, we handle symbolic representations (location granules) at a higher level, from the point of view of the user (the user uses the terminology that s/he requires). Moreover, we concern about how this can enhance the expressivity of location-dependent queries on moving objects.

There are also works that have considered locations at different granularity levels but with no semantic information attached, such as [6]. However, the authors of this work focus on data representation, leaving aside the semantics that we want to add to our granularity model. Their approach is complementary to our work, as ours can be used directly on top of their model.

Finally, existing works on location-dependent query processing implicitly assume GPS locations for the objects in a scenario. As it is difficult to provide a good overview of contributions in this area in a short space, we refer the interested reader to [19]. Although some works acknowledge the importance of considering different location resolutions (e.g., [13]), the processing of classical constraints such as *inside* or *nearest* is not considered in that context. No existing proposal has considered using ontologies to extend the expressivity of the queries dynamically.

7 Conclusions and Future Work

The bare use of granules in location-dependent queries enhances their expressivity. It brings the query to the user's level and may impact not only the query semantics but also the performance and the way the results are presented to

the user. In this paper, we have studied how *semantic location granules* can be used to further enhance the expressivity of this kind of queries. In particular, our proposal:

- provides a definition of *semantic location granules* within the formal framework given by ontologies.
- offers a base ontology to represent the different granularities and the relationships that might exist. This ontology is intended to serve as starting point and can be easily extended to represent the semantics of the relationships that the user might want to add to the system.
- dynamically adapts the query processing to the newly added semantics, allowing to extend the expressivity of the queries that can be processed.

Besides, we have illustrated how this approach allows us to express queries that would not be possible otherwise. As far as the authors know, there is no other proposal that considers the use of ontologies to extend dynamically the expressivity of granule-based location-dependent queries.

As future work, we plan to integrate the proposal into the distributed location-dependent query processing system LOQOMOTION [18], adapting the granule-based processing that it is currently implemented. We are also studying how to extend other constraints with our semantic approach.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Pastel-Scheneider. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press, 2003.
2. T. Bittner, M. Donnelly, and B. Smith. A spatio-temporal ontology for geographic information integration. *International Journal of Geographical Information Science*, 23:765–798, June 2009.
3. G. Cai. Contextualization of geospatial database semantics for human–GIS interaction. *Geoinformatica*, 11:217–237, June 2007.
4. Y. Cai, K. A. Hua, G. Cao, and T. Xu. Real-time processing of range-monitoring queries in heterogeneous mobile databases. *IEEE Transactions on Mobile Computing*, 5:931–942, July 2006.
5. E. Camossi, M. Bertolotto, E. Bertino, and G. Guerrini. Issues on modeling spatial granularities. In *COSIT'03 Workshop: Fundamental Issues in Spatial and Geographic Ontology, Ittingen, Switzerland*. Springer Verlag LNCS, September 2003.
6. E. Camossi, M. Bertolotto, E. Bertino, and G. Guerrini. A multigranular spatiotemporal data model. In *11th ACM Intl. Symposium on Advances in Geographic Information Systems (GIS'03), New Orleans, Louisiana, USA*, pages 94–101. ACM, 2003.
7. S. Dobson. Leveraging the subtleties of location. In *2005 joint conference on Smart objects and ambient intelligence (sOc-EUSAI'05), Grenoble, France*, pages 189–193. ACM, 2005.
8. F. Fonseca, M. Egenhofer, P. Agouris, and C. Câmara. Using ontologies for integrated geographic information systems. *Transactions in GIS*, 6:231–257, June 2002.

9. F. Fonseca, M. Egenhofer, C. Davis, and G. Câmara. Semantic granularity in ontology-driven geographic information systems. *Annals of Mathematics and Artificial Intelligence*, 36(1-2):121–151, 2002.
10. B. Gedik and L. Liu. Mobieyes: A distributed location monitoring service using moving location queries. *IEEE Transactions on Mobile Computing*, 5(10):1384–1402, 2006.
11. T. R. Gruber. Towards principles for the design of ontologies used for knowledge sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation, Deventer, The Netherlands*. Kluwer Academic Publishers, 1993.
12. P. Hitzler, M. Krtzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph. OWL 2 web ontology language primer. W3C recommendation, November 2009.
13. C. Hoareau and I. Satoh. A model checking-based approach for location query processing in pervasive computing environments. In *OTM 2007 Workshops (PerSys'07), Algarve, Portugal*, pages 866–875. Springer Verlag LNCS, November 2007.
14. I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language combining OWL and RuleML. W3C member submission, May 2004.
15. R. Huang and Z. R. Peng. A spatiotemporal data model for dynamic transit networks. *International Journal of Geographical Information Science*, 22:527–545, January 2008.
16. S. Ilarri, A. Corral, C. Bobed, and E. Mena. Probabilistic granule-based inside and nearest neighbor queries. In *13th East-European Conference on Advances in Databases and Information Systems (ADBIS'09), Riga, Latvia*, pages 103–117. Springer Verlag LNCS, September 2009.
17. S. Ilarri, E. Mena, and C. Bobed. Processing location-dependent queries with location granules. In *OTM 2007 Workshops (PerSys'07), Algarve, Portugal*, pages 856–866. Springer Verlag LNCS, November 2007.
18. S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent queries in mobile contexts: Distributed processing using mobile agents. *IEEE Transactions in Mobile Computing*, 5(8):1029–1043, August 2006.
19. S. Ilarri, E. Mena, and A. Illarramendi. Location-dependent query processing: Where we are and where we are heading. *ACM Computing Surveys*, 42(3):1–73, March 2010.
20. J. Schiller and A. Voisard, editors. *Location-Based Services*. Morgan Kaufmann, San Francisco, CA, USA, 2004.
21. G. Stevenson, J. Ye, S. Dobson, and P. Nixon. LOC8: A location model and extensible framework for programming with location. *IEEE Pervasive Computing*, 9:28–37, 2010.
22. M. Vazirgiannis and O. Wolfson. A spatiotemporal model and language for moving objects on road networks. In *7th Intl. Symposium on Advances in Spatial and Temporal Databases (SSTD'01), Redondo Beach, CA, USA*, pages 20–35. Springer-Verlag, 2001.
23. U. Visser, H. Stuckenschmidt, G. Schuster, and T. Vögele. Ontologies for geographic information processing. *Computers & Geosciences*, 28:103–117, February 2002.