On the Applicability of Rule-Based Programming to Location Inference

Katharina Hahn, Kirsten Terfloth, Georg Wittenburg, and Jochen Schiller

{khahn,terfloth,wittenbu,schiller}@inf.fu-berlin.de
Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany

Abstract. Location-based services offer a powerful approach to provide highly relevant information and functionality to a mobile user. Addressing the problem of location inference, we expand the design space by proposing to employ rule-based programming techniques. Based on position coordinates as well as data gathered by environmental sensors, either a precise location or an abstract location class can be deduced. Both types of location may then be utilized to trigger services in an event-centric manner.

1 Introduction

Location-based services (LBS) continue to attract the interest of both academia and industry. The fundamental building block to allow for the usage of LBS on the client side is a subsystem in charge of establishing the current position of a portable device, e.g. by performing a simple mapping from a given tuple of coordinates to a semantically enriched *location*. Data of this kind, along with context- and/or situation-dependent information may be utilized to reason about whether to trigger an LBS of a service provider or initiate service discovery.

A key problem to be solved is to provide a method for specifying what exactly the enriched location, context or situation is. It is worth noting that scenarios are conceivable in which precise information about the current location is not required, since information about the abstract location class is sufficient. Based on the inferred location it must be decided which actions to undertake, i.e. which services to offer to a user, in case a location relevant to the user is detected.

A model on how to define context with the help of sensors on a device and derive a situation by adding semantical information has recently been introduced by [1]. Orthogonally, [2] presents an event-based system with the goal of determining which services are to be taken into account as soon as location information becomes available. The model we present in this paper aims to combine these two complementary features: The FACTS middleware framework [3] can be utilized for both specification and reasoning. Its rule-based programming naturally suits this area due to its inherent event-driven semantics and powerful data abstraction facilities. Additionally, FACTS has been designed with the goal of minimizing resource usage both in terms of memory footprint and processing time. Hence, it is not only a good choice for mobile devices but even more so for highly embedded systems such as wireless sensor nodes.

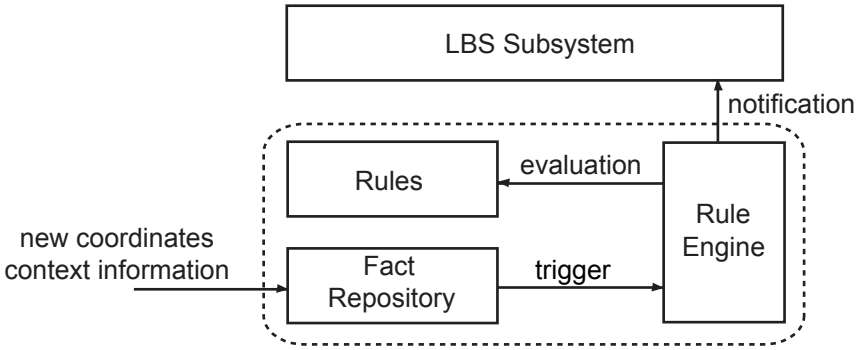

Fig. 1. Interaction between FACTS and the LBS subsystem.

2 A Rule-Based Event-Notification Middleware

The following system model is assumed to allow users equipped with devices that are capable of using wireless communication protocols to employ location-based services: Devices are able to sense their location coordinates, e.g. with a GPS receiver, via GSM cell information, or based on locally deployed RFID tags. Nodes may additionally be equipped with supplementary sensors, e.g. a microphone to detect the noise level or a temperature sensor in order to gather context information of the users current environment.

Using the FACTS middleware, we create a link between LBS, context specification and event notification systems as illustrated in Figure 1. We introduce a hierarchical model of semantically enriched location to aggregate suitable services to sensed context. Those locations may refer to a unique location, e.g. a subway station (*U Hermannplatz, Berlin*) or a specific club (*Goya Club, Berlin*). We also introduce a more abstract notion of location such as a class of location objects, e.g. stations, clubs or universities, that can be used to trigger services or service discovery.

2.1 Precise Location Matching

Initiation of suitable location-based services for a user relies on acquiring knowledge about the current location of the target device. In the following, we assume a location-tuple (longitude, latitude) to be provided by the sensor on the device in question.

The fact repository of the FACTS middleware stores location information concerning *precise location* objects. This precise location object is the result of matching pure geographical data to relevant places, as specified in Listing 1.1. Relevance is defined according to the location-based services which are offered in the environment. Assuming services bound to clubs, the precise location *Goya Club* can be derived from a location information (N52°29.91', E13°21.15') entering the system.

Listing 1.1. Location inference ruleset - Precise location.

```
ruleset LocationBasedClubServices

fact location [long = N52.29.91, lat = E13.21.15, place = "Goya␣Club"]
fact location [long = N52.30.62, lat = E13.24.97, place = "Sage␣Club"]

rule updateLocation 100
<- exists {coordinate}
-> retract {currentLocation}
-> define currentLocation [place = {location place
  <- eval ({this long} == newCoordinateLong)
  <- eval ({this lat} == newCoordinateLat)
}]
-> retract {coordinate}
```

Listing 1.2. Location inference ruleset - Abstract location.

```
fact context [place = "Goya␣Club", type = "night␣club"]

rule updateContext 90
<- exists {currentLocation}
-> retract {currentContext}
-> define currentContext [type = {context type
  <- eval ({this place} == {currentLocation place})
}]
```

2.2 Abstract Location Matching

Either by precise location information or by means of gathering data from additional environmental sensors, so called *abstract locations* can be inferred. These describe the generalization of precise location objects and aggregate features of all derived precise location classes. A rule-based implementation of the process of generalizing an abstract location from a precise location is given in Listing 1.2.

The abstraction class of the example denoted above, is the class *night club*. Services bound to this abstraction class include for example a peer-to-peer dating service which finds matching people within the surrounding according to a predefined profile. The microphone of a mobile node can be used to affirm the context "within a club", and thus the abstract class *night club* if no precise location object is available.

2.3 Inheriting Rules of Super Location Classes

The distinction between precise and abstract location classes is made in order to simplify the coupling of services to locations. All precise location classes inherit the services of their abstract classes. Therefore, it is not required to define rules for several instances which are all sub-nodes of the same abstract class. A sample inheritance hierarchy is depicted in Figure 2.

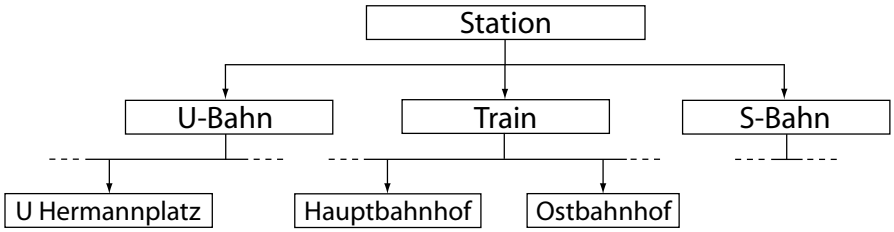

Fig. 2. Location object hierarchy.

When being at a train station in Berlin and therefore within the abstract class *Station*, rules may for example be defined that trigger services to find a ride or to find people who are willing to share their ticket. Within the precise location class *Berlin Hauptbahnhof* additional rules to activate services which display time tables of trains or a map of the station are triggered.

2.4 Service Trigger

By means of the steps defined in the previous subsections, we are able to easily trigger service usage according to the whereabouts of a mobile user. Services are activated through previously defined rules within the ruleset. An application designer can simply add a rule to the ruleset in order to define new triggers for any situation.

3 Conclusion

We propose leveraging the benefits of rule-based programming to ease the task of location inference. To this end, we differentiate between precise and abstract locations depending on available input data, which includes both position coordinates and data gathered by environmental sensors. Upon detecting a location, the middleware may trigger a service in an event-centric way. Adaptations to service notification or expansion of their domain is simply done be adding new rules to the ruleset.

References

1. Meissen, U., Pfennigschmidt, S., Voisard, A., Wahnfried, T.: Context- and Situation-Awareness in Information Logistics. In: Proc. of the International Conference on Extending Database Technology (EDBT'04) Workshops, Heraklion, Crete, Greece (2004)
2. Hinze, A., Buchanan, G.: The Challenge of Creating Cooperating Mobile Services: Experiences and Lessons Learned. In: Proc. of 29th Australasian Computer Science Conference (ACSC 2006), Hobart, Australia (2006)
3. Terfloth, K., Wittenburg, G., Schiller, J.: Rule-Oriented Programming for Wireless Sensor Networks. In: Proc. of the International Conference on Distributed Computing in Sensor Systems (DCOSS'06)/ EAWMS Workshop, San Francisco, USA (2006)