A Model Checking based Framework for Building Correct Context-Aware Systems

Christian Hoareau

Doctor of
Philosophy

Department of Informatics,
School of Multidisciplinary Sciences,
The Graduate University for Advanced Studies (SOKENDAI)

September 2009

Abstract

Context-aware computing refers to the idea that computing devices can sense and react to the physical environment where they are deployed. For example, a context-aware corporate office would intelligently respond to peoples locations and activities by self-adjusting its lighting and temperature automatically, thus reducing its energy footprint.

Context-aware systems model the real world by using increasingly complex and refined contextual-data representations. As a result, their design and management raise several challenges. An important and somewhat unexplored one is to guarantee that context-aware systems correctly capture the intent of their designers once deployed. For example, we might want to assure that the energy conservation of the aforementioned corporate office is actually preserved.

We propose in this thesis a query processing and specification framework that alleviates designing and building context-aware systems. It guarantees the management of contextual information, and can be used for specifying the underlying rules of context-aware systems. Our approach encourages a high-level of abstraction for retrieving contextual information in a robust manner, and supports building "provably-correct" context-aware systems incrementally, by providing modularity and separation of concerns.

The proposed approach aims at complementing existing context-aware services wherein contextual information about the physical and computational environment – information about people, objects, and services – is modeled in a symbolic fashion, and is independent of any particular sensing technology. In current pervasive computing plateform, contextual information and their underlying models are queried an ad-hoc manner. Which makes it impossible to guarantee the quality of the results being returned, and hence the reliability of context-aware services.

The main idea behind our framework is to apply and adapt the principles of model checking to query the contextual data structures. Because such query mechanisms have to be sound, our approach is build upon a logic-based query language. We therefore ensure that the results of any query (i) do not miss any information that satisfy its necessary and sufficient conditions and (ii) do not contain any information that does not satisfy the conditions. We describe the implementation of our framework and discuss its applicability to existing graph-based contextual models.

List of Publications

Journals

- Christian Hoareau and Ichiro Satoh, *Modeling and Processing Information for Context-Aware Computing: A Survey.* New Generation Computing, 2009. (To appear)

- Christian Hoareau and Ichiro Satoh, *Query Language for Location-Based Services: A Model Checking Approach.* IEICE Transactions on Information and Systems, Special Issue on Knowledge-Based Software Engineering, Vol.J91-D, No.3. April 2008.

Conferences and Workshops

- Christian Hoareau and Ichiro Satoh, *From Model Checking to Data Management in Pervasive Computing: A Location-based Query-processing Framework.* Proceedings of the ACM International Conference on Pervasive Services (ICPS'09), July 2009.

- Christian Hoareau and Ichiro Satoh, *Logic-Inspired Query Processing Framework for Sensor Networks.* Demonstration Session of the 5th International Conference on Networked Sensing Systems (INSS'08), June 2008.

- Christian Hoareau and Ichiro Satoh, *Hybrid Logics and Model Checking: A Recipe for Query Processing in Location-Aware Environments.* Proceedings of the IEEE 22nd International Conference on Advanced Information Networking and Applications (AINA 2008). March 2008.

- Christian Hoareau and Ichiro Satoh, *Logic-Inspired Query Processing Framework for Pervasive Computing.* Demonstration Session of the 6th IEEE International Conference on Pervasive Computing and Communications (PerCom'08). March 2008.

- Christian Hoareau and Ichiro Satoh, *A Model Checking-Based Approach for Location Query Processing in Pervasive Computing Environments.* Proceedings of the 2nd OTM International Workshop on Pervasive Systems (PerSys'07). Nov. 2007.

Contents

List of Figures

List of Tables

The most profound technologies are those that disappear.

– Mark Weiser

1 Introduction

Context-awareness is embedded into the fabric of our human nature. Consciously or unconsciously, we often derive our actions and behaviors from a particular set of circumstances, the *context*. This context is rich, subjective and ever-changing. The actual meaning of one's context, and thus its relevance, depends on what activity one might be engaged in. Context is multi-dimensional; it can encompass perceptual information – environmental (e.g., the level of pollution), physical (e.g., one's current location), social (e.g., one's family and colleagues), or temporal (e.g., the time of the day), just to name a few. Non-perceptual information, like memories of one's past experiences or one's emotional state, is part of one's context as well.

For one thing, our context-awareness stems from a huge number of stimuli coming from our surrounding world. These stimuli are first captured by an extremely sophisticated sensory system, encoded by our brains into electrical signals, then processed

in one way or another, stored for later retrieval, and/or eventually lost. This process gives us the ability to sense, and adapt our behaviour to, the world around us. We are context-aware beings, and we would like to see our ever-growing ubiquitous computing ecosystem – homes, cities, personal and wearable devices – become context-aware as well. Ultimately, the technology will adapt to us.

The idea of context-aware computing was introduced in some of the pioneering work on ubiquitous computing research and has been subject to extensive research since. Context-aware computing stems from the vision articulated by Marc Weiser [2] in his seminal 1991 paper: "The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it". The goal of context-aware computing is to acquire and utilize information pertaining to the physical world, and then select, configure, and provide a variety of services accordingly.

Many researchers have explored the data representation of contextual information and proposed various *context models*, including *location models*. Location models are usually updated by tracking (or positioning) sensing systems located throughout the environment (see Fig. 1.1). They maintain the representations of users and objects according to the data measured by such systems and provide mapping between the physical and virtual worlds. However, the focus of current research on context modeling has been on the representation and maintenance of context-aware information. Instead, mechanisms to retrieve or query the information maintained in context models have attracted scant attention. This is a serious obstacle to the advance and growth of pervasive computing, including context-aware computing. In fact, a large number of context models have been explored thus far, but most of these do not offer any support for query processing. Also, the few models with built-in query processing rely on ad-hoc mechanisms to access contextual information, which makes it difficult to guarantee the quality of the results.

In the database research community, a data model is a collection of conceptual tools that are used to model the representation of real-world entities and the relationships among them [3]. There has been a broad consensus about what these conceptual tools

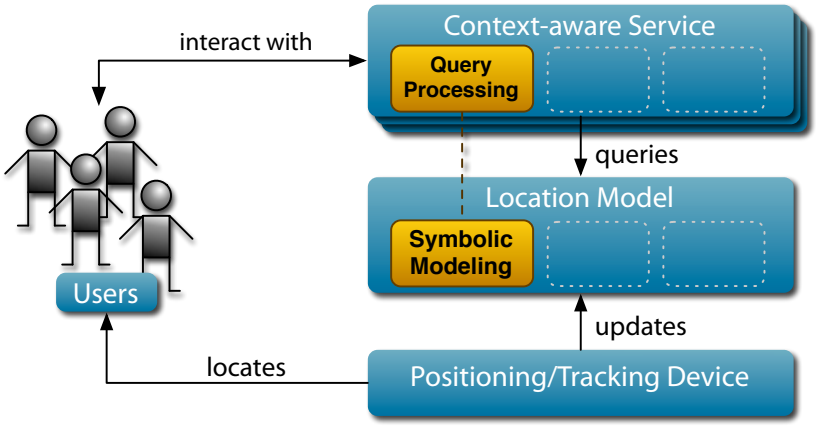

FIGURE 1.1: Overview of a context-aware system.

should offer, i.e., data structuring, description, maintenance, and some mechanisms to retrieve or query the data [4]. However, query mechanisms for existing database systems are not available in context-aware computing, because they do not support the notion of context awareness. Therefore, pervasive computing needs query mechanisms that are designed for context-aware computing.

We propose a logic-based framework to address these issues. It is designed to support, and benefit from, *existing* symbolic location models, which are the standards for representing context of the physical world. Our contribution is emphasized in Fig. 1.1. The basic idea behind the framework is to apply model-checking principles to query the underlying data structures. Our approach is build upon a logic-based query language. Indeed, such query mechanisms must to be sound like SQL, which is based upon relational algebra. We presented the formal definition and semantics of an earlier version of the language in an earlier paper [5], which lacked a description of the implementation. We would like to emphasize that our framework is aimed at complementing existing location models. That is, the language itself can be applied to other existing symbolic location models, although it was initially designed to support hierarchical-based location model.

Part I

Research Domain and State of the Art

The reasonable man adapts himself to the world ; the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man.

– George Bernard Shaw, Maxims for Revolutionists : Reason, Man and Superman (1903)

2

Research Domain

CONTEXT-AWARE COMPUTING has enjoyed remarkable attention from researchers in diverse areas such as mobile computing (Schilit et al., 1994) and human computer interaction (Moran & Dourish, 2002). It is also an important idea explored in connection with pervasive computing and ambient intelligence.

The notion of context itself is not new and has been explored in areas such as linguistics, natural language processing, philosophy, AI knowledge representation and problem-solving, and theory of communication (McCarthy, 1993; Akman, 2002; Bouquet et al., 2003; Brezillon, 2003). In such work, context is given focus and primacy (e.g. treated as first-class objects in a logic), enabling assertions to be made about contexts and context to be explicitly reasoned about in applications.

The Free On-line Dictionary of Computing defines context as "that which surrounds, and gives meaning to, something else". Such a definition might be instantiated

according to the need. Whether that "something" is an assertion in a logic, an utterance, or a computer system, with an appropriate definition for meaning, the intuition captured by the word "context" serves its purpose. The work by Schilit et al. (1994) provides an instantiation of that definition, from the perspective of distributed, mobile, and ubiquitous computing (or pervasive computingone view of pervasive computing is as a combination of mobile computing and ubiquitous computing): a person is that something and context refers to information about a persons proximate environment such as location and identities of nearby people and objects. In Dey (2001) is an operational (and arguably broader) definition of context:

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

Context-aware applications aim to use such contextual information to do the right thing at the right time automatically for the user. There has been much work in identifying what such information can be, the structure of the information, how to represent such information, and how to exploit it for a specific application. Such work might focus on a specific kind of contextual information such as location models,4 world models (e.g. Lehmann et al., 2004) and activity models (e.g. Koile et al., 2003; Muhlenbrock et al., 2004; Tapia et al., 2004), where activity typically refers to some action or operation undertaken by a human being, such as bathing, doing laundry, toileting, preparing breakfast, and listening to music, and so differs from situation. Perhaps one could conceive of a person in the state of preparing breakfast as a situation. However, in general, activity and situation are clearly not interchangeable, and we consider activity as a type of contextual information which can be used to characterize the situation of a person (e.g. preparing breakfast means the person is busy or has just woken up), or identify characteristics of contextual information (Henricksen et al., 2004).

Pervasive computing utilizes contextual information about the physical world. Hence, the connection of sensor information to context-aware pervasive computing is clearly important (Hopper, 1999; Yoshimi, 2000; Barkhuus, 2003; Patterson et al., 2003), and

relates to what can be sensed, the best way to acquire sensor information, and how to reason with sensor information to infer context. In fact, any information which can be practically obtained via sensors can be used as context, including the emotional states of users (Picard, 1997) and movements (Headon, 2003). When the entity is an artifact instead of a person, we have context-aware artifacts.

There is tremendous variety and diversity in what can be context, and the way context can be acquired and modelled, and this is an avenue of much interest and research. Given the challenges in representing, structuring, managing and using context, it is not surprising that various knowledge representation formalisms and techniques have been applied, ranging from ontologies (Matheus et al., 2003; McGrath et al. 2003; Chen et al., 2004b; Wang et al., 2004) (that can provide concepts for describing context and enable reasoning with and reuse of contextual information), first-order logic theories (Katsiri & Mycroft, 2003; Ranganathan & Campbell, 2003), to conceptual graphs (Peters & Shrobe, 2003). Such work, however, is not simply a return to previous AI knowledge representation about context, but consider what aspects of the physical world should be sensed for a given application and how best to represent such aspects, how to reason with sensed information, and the software engineering of context-aware pervasive systems.

Related to the notion of context is the notion of situation. The relationship between context and situation is illustrated in the above operational definition. A definition of situation from the American Heritage Dictionary is as follows: "The combination of circumstances at a given moment; a state of affairs". Besides context, Dey (2001) also defines situation as "a description of the states of relevant entities". Hence, the idea is of aggregating (perhaps varieties of) context information in order to determine the situation of the entities (relevant to an application). In this sense, the situation might be thought of as being at a higher level of abstraction than context.

Learning would be exceedingly laborious, not to mention hazardous, if people had to rely solely on the effects of their own actions to inform them what to do. Fortunately, most human behavior is learned observationally through modeling.

– Albert Bandura

3

Context Modeling

Contents

CONTEXT-AWARE SYSTEMS are concerned with the acquisition of context (e.g., using sensors to perceive a situation), the abstraction and understanding of context (e.g., matching a perceived sensory stimulus to a context), and application behaviour based on the recognized context (e.g., triggering actions based on context). Context-awareness is regarded as an enabling technology for ubiquitous computing systems. While there seems to be a consensus on the importance and usefulness of capturing and processing contextual information, its very nature, scale, and complexity poses major challenges [6]:

- **Acquisition of context**: Contextual information is acquired from non-conventional and heterogenous sensors, which may be connected through a network.
- **Abstraction of context**: It should be abstracted, because it may depend on the sensors that acquire it [7].
- **Understanding of context**: It needs to be represented, managed, and used in relevant data structures and algorithms to be processed by context-aware services.

Of these technical issues, most researchers have addressed the third. In fact, several international workshops have been dedicated to these very issues in recent years[1]. This chapter reviews related work that has focused on contextual information processing and management in addition to context modeling. We focus on the conceptual tools that have been proposed to address pertaining to query processing. We overview the most relevant approaches to modeling context for ubiquitous computing. We provide an analysis of the existing surveys that have focused on context modeling, and discuss the relevance of their underlying evaluation frameworks (Sect. 3.3). We place particular emphasis on the query-processing approaches used to manage contextual information

[1]Workshop on Modeling and Reasoning in Context (MRC 2008 - http://events.idi.ntnu.no/mrc2008/). Workshop on Context Modeling and Reasoning (CoMoRea 2009 - http://nexus.informatik.uni-stuttgart.de/COMOREA/). Workshop on Context Modeling and Management for Smart Environments (CMMSE'08 - http://flash.lakeheadu.ca/ rbenlamr/cmmse08/)

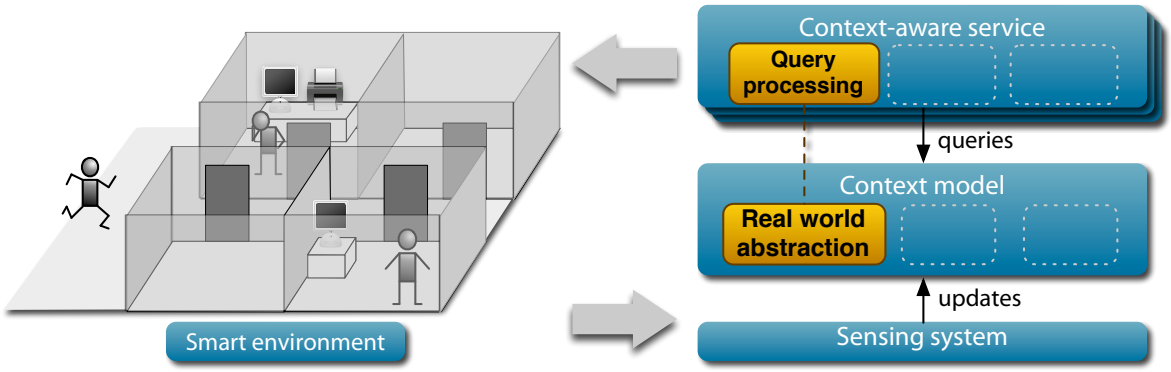

Figure 3.1: Context-aware services can sense and react to the physical environment where they are deployed.

(Sect. 4.2). We discuss various problems, shortcomings, and challenges posed by context modeling at large, and highlight some proposals to address some of them (Sect. 4.3).

Other important topics that are not directly related to computer-system issues fall outside the scope of this survey. Some of these are context-awareness applied to the fields of human-computer interactions, expert systems, and software agents. As a side note, we have not specifically overviewed context-aware systems that have been developed since their introduction in the early 90s, nor have we looked at context middleware, infrastructures and toolkits. We refer to Strimpakou et al.'s review [8] of the main projects and to Baldauf et al. [9] for their comprehensive overviews.

Much of the context information involved in pervasive systems is derived from sensing devices. There is usually a significant gap between sensor output and the level of information that is useful to applications. Therefore, this gap must be bridged by various kinds of processing of context information before the information is passed to context-aware services.

3.1 Location Models

Location is one the most typical contextual information. Context-aware applications primarily use location – of people, objects, and computational devices – as their main

source of contextual information. Indeed as significant progress is achieved in location-sensing and -tracking technologies – e.g., the Global Positioning System (GPS), Radio Frequency IDentification (RFID) tags, and ultrasonic-based tracking systems – location has become de-facto the preeminent contextual information. The functions of a location sensor can be classified into two typical approaches: positioning and tracking. The former measures it own location with the help of the infrastructure, and the latter measures the location of other located objects. Their output may be in raw coordinates, whereas an application might be interested in identifying the building or room users are in. Moreover, requirements can vary between applications. Therefore, a context model must support multiple representations of the same context in different forms and at different levels of abstraction, and must also be able to capture the relationships that exist between alternative representations.

There are many types of location information, e.g., rooms in a house and latitude-longitude on the earth. These different types need to be expressed with different coordinates and data structures. Most existing context-aware systems maintain contextual information in the output format of the underlying sensing system. Therefore, they tend to depend on their current sensing systems. They cannot be used when their sensing systems are changed. Therefore, several researchers have proposed and defined data structures for abstracting and modeling location-based information to be as independent of sensing systems as possible. Becker and Dürr reviewed the location models [10].

3.1.1 Geometric Models

They represent the positions of people and objects as geometric coordinates, captured by positioning sensors. The most prominent reference-coordinate system for outdoor environments is GPS, which is widely used for navigation-based applications. However such fine-grained information is often meaningless in human interactions and lacks any semantics for describing the relations between locations. Geometric coordinates are therefore contextualized with secondary human-readable information. In navigation-based applications, GPS coordinates are represented by moving points on city maps.

3.1.2 Symbolic Models

They are applied in areas where information about data interconnectivity or topology is more important, or as important, as the data themselves. In these applications, the data and relations between them, are usually at the same level. Introducing graphs as a modeling tool has several advantages for these types of data. It allows for data to be more naturally modeled. Graph structures are visible to the user and they allow a natural way of handling application data, e.g., hypertext or geographic data. Graphs have the advantage of being able to keep all the information about an entity in a single node and show related information by arcs connected to it. Existing symbolic-based models can be viewed as subsets of this category.

3.1.3 Hybrid Models

We need to choose location models carefully with respect to the requirements for spatial reasoning and modeling effort involved. Both humans and computers can easily understand the symbolic model, but there is no precision with geometric models. Therefore, several researchers have proposed eclectic models, called *hybrid location models* [11]. A hybrid model has both the advantages and disadvantages of the previous tow models. A few researchers have proposed semantic models, which have rather been like focusing purely on position by using ontologies.

3.2 Context Models

While the computer-science community has initially perceived context as a matter of user location, as Dey stated [12], in the last few years this notion has not simply been considered as a state, but part of a process in which users are involved; thus, sophisticated and general context models have been proposed to support context-aware applications that use them to (a) adapt interfaces, (b) tailor the set of application-relevant data, (c) increase the precision of information retrieval, (d) discover services, (e) make user interactions implicit, or (f) build smart environments [13]. For example,

a context-aware mobile phone may know that it is currently in a meeting room, and that its carrier has just sat down. The phone may conclude that the user is currently in a meeting and reject any unimportant calls [14].

3.2.1 A Great Variety of Frameworks

Computing systems may need to understand the real world to provide services according to context within it. But unlike humans, they cannot maintain all the information about the real world. Consequently, they are required to extract and maintain their context of interest as models of the real world inside them. Most existing context models have been designed and implemented in an ad-hoc manner, in the sense that their context models are premature or dependent on the underlying sensing systems. A few researchers have begun to look at the frameworks for context-aware systems more generally, independently of specific applications, including context middleware, infrastructures, and toolkits [15] [16] [17] [18]. Such work facilitates the building of context-aware systems. Tools for end-users to program context-aware systems are also being built on top of these infrastructures.

Chen and Kotz [19] thoroughly reviewed the research on context-aware mobile computing. They provided a complete overview of the major context-aware applications that had been built, and gave a snapshot of the underlying context models. They highlighted the following six data structures.

Key-value Model

A key (or identifier) in this simple model corresponds to an attribute of the environment that has a value. This value is usually measured by sensors embedded in the environment. For example, `<Room 31 temperature, 24>` is such a `<key, value>` pair. Key-value models were used by Schilit et al. [20] to manage location information. For obvious reasons, this simple model has been replaced by more expressive and flexible alternatives. The model has problems with its expressiveness but it can be directly maintained on large-scale distributed systems like cloud computing, where

data are maintained on a huge number of computers that are loosely coupled.

Markup Model

Markup-based models use a hierarchical data structure consisting of markup tags with attributes and content. Profiles represent typical markup-scheme models. Typical examples are the Composite Capabilities/Preference Profile (CC/PP) and User Agent Profile (UAProf) which are encoded in RDF/S with XML notation [21].

Object-oriented Model

This is based on the concept of objects and relationships between them as in the object-oriented programming paradigm. Rather than using implementation-based concepts like records, object-oriented models provide flexible structuring capabilities by fully-fledged object-orientation mechanisms such as encapsulation, reusability, and inheritance. Objects are used to represent various types of contextual information (e.g., temperature and location); they encapsulate the operations used for context processing.

Logic-based Model

Facts, expressions, and rules are used to define a context model. Contextual information comprise facts described by some rule-based mechanisms. Ranganathan et al. [22] and Katsiri and Mycroft [23] applied first-order logic to reason with contextual information, and Henricksen [24] applied the same techniques to describe and reason with situations. Others like Sohn [25] provided end-users with a toolkit that allowed them to develop context-aware applications by specifying rules. Overall, rule-based programming has proved to be intuitive and well-suited to prototyping context-aware applications.

Ontology

Numerous projects use ontologies and the tools from the “Semantic Web” to represent and reason with context [26]. Ontology-based techniques support a vocabulary for

situation predicates [27], so that the data representations in these projects might be shared across different systems, or even retrieved from a store over the Web. Also, ontologies can also provide vocabularies and additional semantics to sensor predicates, i.e., such predicates can represent context attributes specified by an ontology. This approach not only has the advantages but also the disadvantages of ontology-based data representation.

Situation Logic

Loke [27] took rule-based programming further by representing situations and not only programming rule-based triggers for context-aware actions. He explored the integration between context infrastructures and LogicCAP, with the infrastructures providing sensed contextual information and LogicCAP as the programming layer.

3.3 Evaluating Context Models

This section reviews some existing survey papers and discusses their methods to classify and evaluate context models. These papers are presented in chronological order of publication.

3.3.1 Requirement-based Evaluation Framework

In a 2004 paper, Strang and Linnhoff-Popien [1] evaluated six modeling approaches that are deemed to be the most relevant to context-aware computing. Their evaluation framework was based on a set of six requirements that ubiquitous computing systems should satisfy.

Distributed Composition. Context-aware systems are distributed, and thus do not have any central instance responsible for the creation, deployment, or maintenance of data and services, particularly context descriptions. Instead, the composition and administration of a context model and its data varies with notably high dynamics in terms of time, network topology, and source.

Partial Validation. On the structural as well as on instance level, even if there is no single place or point in time where the contextual knowledge is available on one node as a result of distributed composition. This is particularly important because of the complexity of contextual interrelationships, which make any modeling intention error-prone.

Richness and Quality of Information. The quality of information delivered by sensors varies over time, as well as the richness of information provided by different kinds of sensors characterizing an entity in an ubiquitous computing environment, may differ. Thus, a context model appropriate for use in ubiquitous computing should inherently support both quality and richness.

Incompleteness and Ambiguity. The set of contextual information available at any point in time characterizing relevant entities in ubiquitous computing environments is usually incomplete and/or ambiguous, particularly if this information is gathered from sensor networks. This should be covered by the model, for instance by interpolating incomplete data on the instance level.

Level of Formalism. It is always a challenge to describe contextual facts and interrelationships in a precise and traceable way. For instance, carrying out the task "print document on the nearest printer" requires a precise definition of terms used in the task, for instance what "nearest" means. It is extremely important that all the entities share the same interpretation of the data that are exchanged, as well as their underlying meaning.

Application to Existing Environments. From the implementation perspective, it is important for a context model to be applied within an existing infrastructure of ubiquitous computing components.
In addition to the models previously described, Strang and Linnhoff-Popien added the graphical models. Their conclusions are summarized in Fig. 3.2. The results reveal that ontologies have the most expressive models and fulfil most of the requirements.

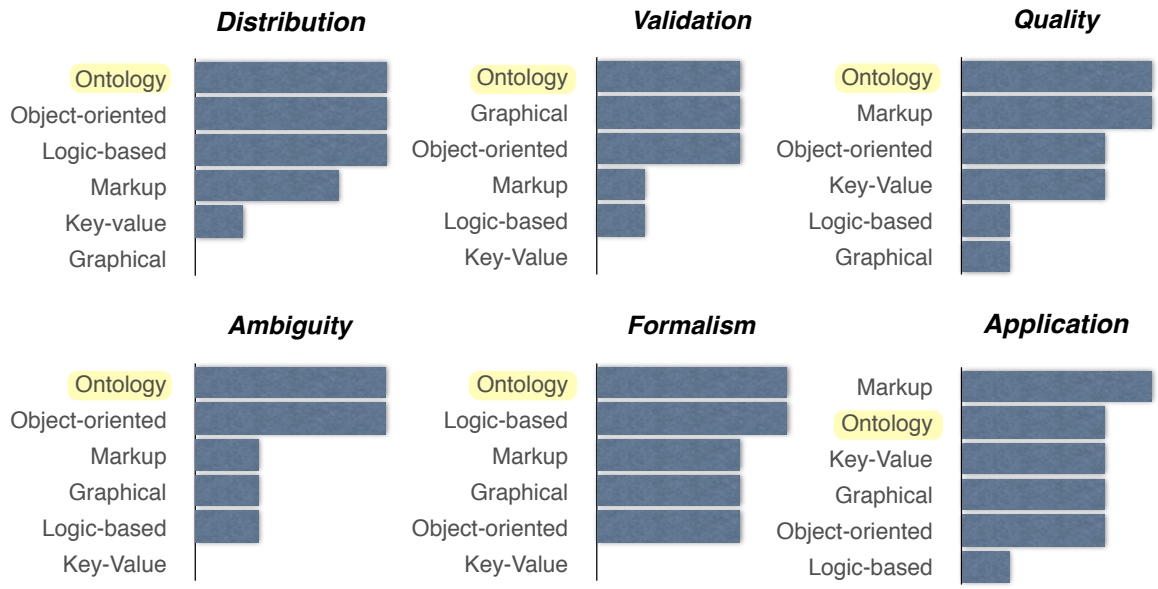

FIGURE 3.2: Evaluation of the six major context-modeling approaches based on six requirements defined by Strang et Linnhoff-Popien [1].

3.3.2 Data-based Evaluation Framework

Bolchini et al. [13] provided a comprehensive evaluation framework for comparing context-aware services and their underlying context models. The results of their analysis were intended to help designers build context-aware systems and choose the context model that was the most appropriate to their systems' specifications and requirements. They classified the features of context models into three groups:

- Modeled aspects: The set of context dimensions managed by the model. This includes the notion of space and time – which can be represented as absolute or relative –, context history, subject, and user profile.

- Representational features: The general characteristics of the model itself. This includes the type and level of formalism, flexibility of the model, variable context granularity, and valid context constraints.

- Context management and usage: The way the context is built, managed, and exploited.

Bolchini et al. placed great emphasis on context-aware data-tailoring, and designed their evaluation framework according to this. Contextual data acquired from the sensing environment was used to filter the data normally processed by the applications. Context-data tailoring has three goals: (i) to provide more relevant data for users (e.g., time- and location-based), (ii) to match the physical constraints of the devices, and (iii) to improve the efficiency of query processing.

3.3.3 Discussion

Any context model is designed according to a specific set of requirements. These requirements are based on the nature of the contextual information to be handled by the context-aware services. In this section, we look at the characteristics of context information, and then discuss the requirements used to build the evaluation frameworks that were previously presented.

Target-dependent Context Information

Contextual information can have different natures. Dey [12] distinguished between three kinds of contextual information or entities: places, people and things. Each entity is characterized by attributes that fall into one the following categories: identity, location status, or time. Context information can be characterized as static or dynamic. Static context information describes those aspects of a pervasive system that are invariant, such as a persons birthday.

The persistence of dynamic context information can be highly variable. E.g., the relationships between colleagues typically can endure for months or years, whereas a persons location and activity often change from one minute to the next. Persistence characteristics influence the means by which context information must be gathered. While it is reasonable to obtain largely static context directly from users, frequently changing contexts must be obtained by indirect means, such as through sensors.

Pervasive computing applications are often interested in more than the current state of the context. They can also rely on activities planned for the future. As a result, the

contextual description might comprise context histories, both past and future.

Requirements of Services

Arguably, distributed composition is orthogonal to any context-modeling issues, because it is concerned with the architecture of the system and its underlying implementation. Also, the richness and quality of the information is both technology- and application-dependent. Data models have their own sets of requirements. For example, Korpip and Mntyjrvi [28] presented four requirements and goals having designed a context ontology:

- Simplicity: The expressions and relations should be as simple as possible to simplify the work done by applications developers.
- Flexibility and extensibility: The ontology should support the simple addition of new context elements and relations.
- Genericity: The ontology should not be limited to special kinds of context atoms but rather support different types of contexts.
- Expressiveness: The ontology should allow as many context states to be described as possible, and with arbitrary levels of details.

Knowledge- vs. Data-Modeling

While summarizing Strang and Linnhoff-Popien's review [1] of the major context-modeling approaches, Moore et al. [29] raised somes concerns about knowledge-based modeling, i.e. ontology-based modeling. In particular, they pointed out that the modeling function of a given ontology fails to address the issue of context processing. Acknowledging that ontologies provide a powerful support for knowledge reasoning, Roussaki et al. [30] nevertheless mentioned the same shortcomings in regard to the acquisition, management, and processing of contextual data. Serious concerns about ontologies have been raised [31] in the Semantic Web community as well. Pils et al. [32] summarized these problems in terms of:

- Readability: Because of the strict standardisation of semantics, it is difficult for us to comprehend specifications defined by an ontology, and find relevant pieces of information.

- Costs: It can be expensive to adopt ontologies especially for small businesses like shops or restaurants.

- Disorder: The more the ontology model is structured, the more effort is required to increase its structure or even maintain the status quo. Indeed, the system $\mathcal{S}$ of semantic specifications exhibits the entropy $\mathcal{H}$: $\mathcal{H}(\mathcal{S}) = -\sum_{i=0}^{n} \pi(l_i) \cdot \log(\pi(l_i))$ where $\pi(l_i)$ is the probability that humans will misinterpret information, and it is proportional to the frequency of information use.

3.3.4 Context Modeling as Active Database Systems

Conventional databased management systems are passive. Data are created, retrieved, modified, and deleted only in response to operations issued by users. By contrast, context-aware computing is required to automatically carry out some services in response to certain changes in the real world, once conditions are being satisfied. Therefore, it needs some facets that active database systems have [33]. Unfortunately, there have been few practical implementations of general-purpose active database systems [34], and as a result, several research projects on context-aware computing have attempted to implement active-database systems optimized for context-aware computing or to use various techniques found in the literature on active-database systems.

Satoh [7] [35] introduces a world model for location-aware and user-aware services in ubiquitous computing environments, that can be dynamically organized like a tree based on geographical containment such as that in a user-room-floor-building hierarchy. Each node in the tree is constructed as an executable software component, that enables location-aware services to be managed without databases and by multiple computers. The proposed world model also provides a unified view of the locations, as they not only refer to physical entities and spaces, including users and objects, but also computing devices and services. The model is unique among other existing

context-aware computing systems, because it consists of not only data elements but also programmable entities to define services. Therefore, the model itself automatically provides context-aware services in response to structural changes in the model corresponding the contextual changes in the real world.

To know an object is to lead to it through a context which the world provides.

– William James

4

Context Processing

Contents

CURRENT RESEARCH on ubiquitous computing has paid attention to the design and implementation of application-specific location-aware services, such as smart rooms and navigation systems. These have often been designed for particular sensing systems (e.g., GPS and RFID tags) that capture context information about users and objects in the environment. In this section, we review work pertaining to the management of contextual data, which is often overlooked. In fact, most existing context-aware services maintain and process contextual information in an ad-hoc manner and/or rely on centralized and inadequate data infrastructures. Hereafter, we present the underlying connection between context models and query processing.

From a data-management viewpoint, contextual information is data that have to be captured, processed, and managed by the system. We previously discussed typical context models such as key-value and markup models. These models may be suitable for expressing contextual information, but not so for maintaining the information in computing systems. Context modeling shares very similars characteristics with data modeling. There has been a broad consensus in the database and data-engineering communities about the gap between the structure of data modeling and structure of data maintenance.

4.1 Mapping Contextual Information on Relational Database Model

A variety of data structures has been proposed and these have then faded away. Of these, the relation database (RDB) model, which was proposed by Edgar Codd in 1970 [36], is still the reference model in database systems. Indeed, there have been many commercial and open-source implementations of RDB. Many researchers have attempted to maintain contextual information in existing RDB systems. However,

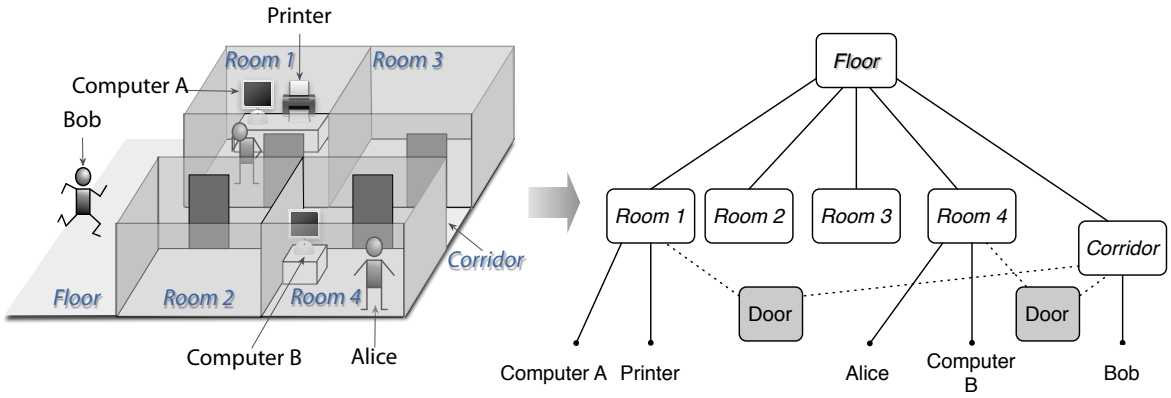

Figure 4.1: Example of a graph-based data structure representing a real-world space.

the RDB model is different from context models. Nevertheless, several researchers have extended RDB systems to have the ability of supporting contextual information. Spatial databases represent the most typical extensions of RDB systems [37]. They can store and query data related to objects in space, including points, lines and polygons, whereas typical RDB systems only support numerical and character types of data. They may be useful for maintaining geometric models for location-awareness, but they cannot support other models.

4.1.1 Resurgence of Graph-database Models

Graph database models are applied in areas where information about data interconnectivity or topology is more important, or as important, as the data itself [4]. In these applications, the data and relations between the data, are usually at the same level. Graph database models are often used in knowledge representations such as semantic nets and frames. Introducing graphs as a modeling tool has several advantages for this type of data:

1. It allows for a more natural modeling of data. Graph structures are visible to the user and they allow a natural way of handling applications data, e.g., hypertext or geographic data. Graphs have the advantage of being able to keep all the information about an entity in a single node and showing related information by arcs connected to it. Graph objects (like paths and neighborhoods) may have

first order citizenship; a user can define some part of the database explicitly as a graph structure, allowing encapsulation and context definitions [38].

2. Queries can refer directly to this graph structure. Associated with graphs are specific graph operations in the query language algebra, such as finding shortest paths and determining certain subgraphs. Explicit graphs and graph operations allow users to express a query at a high level of abstraction. To some extent, this is the opposite of graph manipulation in deductive databases, where fairly complex rules often need to be written. It is not important to require full knowledge of the structure to express meaningful queries. Finally, for purposes of browsing it may be convenient to forget the schema.

3. For implementation, graph databases may provide special graph storage structures, and efficient graph algorithms for accomplishing specific operations.

4.1.2 Semi-structural Data Models for Contextual Information

There has been an avalanche of work on *semi-structured* data in the last decade in database engineering research, because XML, which is a data representation based on a semi-structured model has been widely used. Several researchers have attempted to represent contextual information using XML-based notation, in particular the RDF (Resource Description Framework) notation. RDF is a standard put forward by the the World Wide Web Consortium (W3C) to mainly represent metadata on Web data. RDF represents data resources with object-attribute-value groups. It provides a means of adding semantics to a document without making any assumptions about its structure. Therefore, RDF can support the complicated data structures of contextual information. In fact, there have been many projects that have attempted to represent contextual information in XML or RDF. However, XML and RDF notations are not suitable for maintaining contextual information. Contextual information dynamically evolves according to changes in the real world, whereas the notations are static.

4.2 Query for Contextual Information

Query mechanisms are indispensable for most database systems [3]. There has been a broad consensus about what these conceptual tools should offer: data structuring, description, maintenance and some mechanisms to retrieve or query the data [4]. However, research on contextual information has focused on the representation of information. Instead, few researchers have paid attention to query mechanisms for contextual information. This is a serious obstacle in context-aware services.

4.2.1 Hybrid Approaches

Aiming at providing contextual-information retrieval that is both scalable and robust, Roussaki et al. [30] enhanced the database with some location-based retrieval mechanisms. They implemented a hybrid context model that maintained (i) symbolic entities like streets and buildings, and (ii) objects located in a coordinate system. The context database maintained a hierarchy of entities. By combining the semantical expressiveness of ontologies with the hierarchical structure of tree-based location models, they could dispatch and route the queries along the context database hierarchy and obtain a significant increase in information retrieval. The same approach has been developed further to improve context filtering [32].

Ilarri et al. [39] investigated query processing for location-based services. They targeted outdoor applications that have to handle a fair number of moving objects in real-time. Responsiveness and performance are therefore critical. The authors focused on geometric location models that provide both quantitative and high-resolution data on moving objects. Acknowledging that fine-grain GPS coordinates may be inappropriate for many location-based applications (high-resolution requires more overhead), they introduced the notion of location granules. Granules are geographical areas that can be defined at different scales, e.g., freeways and buildings. However, the semantics of the granules are rather poor, since relationships between granules are not explicitly defined. Therefore, the query language used to retrieve the location cannot benefit from the spatial semantics of the granules.

4.2.2 Application-independent Query Processing

Grossniklaus and Norrie [40] proposed an object-oriented version control to address the challenges of data management in context-aware services. The query-processing framework was application-independent. Indeed, as many frameworks were specific to a single application domain, their notion of context (in particular the context dimensions e.g., users, devices, and environmental factors) was only relevant to a subset of applications. Moreover, representation and storage are often implied by the context model. They proposed a general approach, not limited to any particular context model. It was similar to Bolchini et al.'s [13] that emphasized the importance of context-based data tailoring.

Perich et al. [41] proposed a solution to managing data in pervasive computing applications. It broadly consisted of two parts: treating the devices as semi-autonomous entities guided in their interactions by profiles and context, and designing a contract-based transaction model. The profile was grounded in a semantically rich language for representing information about users, devices, and data objects each described in terms of "beliefs", "desires", and "intentions" - a model that has been explored in multi-agent interactions. They introduced data-based routing algorithms, semantic-based data caching, and replication algorithms enabling mobile devices to utilize their data-intensive vicinities.

4.2.3 Application of Formal Methods

Hoareau and Satoh [5] introduced a query-processing framework for location-based services that is based on model checking. As most context-aware systems tend to query the underlying location models in an ad-hoc manner, it is difficult to guarantee the quality of the results and the reliability of context-aware services. Hoareau and Satoh proposed (i) a hybrid logic-based language that can express location queries over symbolic representation of space, and (ii) a model-checking-based query-processing engine that processed queries over these symbolic representations. They subsequently implemented the language in to a query-processing framework. The latter ensures

that the results of any query (i) would not miss any information that satisfied its necessary and sufficient conditions and (ii) would not contain any information that did not satisfy the conditions. Model checking has proven valuable to query graph-based context models. Indeed, query language for context-aware services need some theoretical underpinnings, just like SQL is built on relational algebra.

4.3 Discussion

4.3.1 Holistic Approach to Context Awareness

Davies and Gellersen [42] discussed the technological and sociological challenges in creating the widely deployed, ubiquitous computing systems that Weiser envisioned more than fifteen years ago. They considered Weiser's scenario [2] and its "foreview mirror" application, which Sal uses on her way to work, and describe how it could be implemented with *existing* technologies. The application would require a satellite navigation and information system, a video camera-based system for detecting available parking spaces, and a location-aware system for recommending nearby shops. But since these three components are currently designed, built, and deployed *independently*, they lack the actual mechanisms for seamlessly working together. In particular, they cannot integrate and process different views of the world, or contexts, in which they run.

Arguably, one needs to have a holistic approach to context-awareness. Such an approach, however, raises several issues that go beyond mere engineering. We will not elaborate on these issues, but it is worth mentioning that besides people's concerns about their privacy, the designers of context-aware systems may have to cope with various legislations on data protection, which could eventually inhibit the deployment of their systems. Moreover, the question of cost repartition is critical for the success of context-aware systems, in cases where several companies would be providing components for *individual* services. As Davies and Gellersen [42] point out, no effective business models have been successfully implemented thus far.

4.3.2 Semantic Computing

The technical issues of integrating different components for the sake of seamless and more enjoyable user experiences have been initially approached within the Web community. The *Semantic Web*, which Noy defines as "a form a Web content that will be processed by machines with ontologies as its backbone" [31], has had a considerable influence on the context-aware research community. For example, in their review on context modeling, Strang and Linnhoff-Popien [1] concluded that ontologies were the most flexible and effective approach. The Semantic Web stems from our sheer inability, as humans, to process the ever-growing amount of online information. We need some help from machines. The idea behind the Semantic Web is to create *independent* software agents that would share the notions they operate with. This mutual "semantic foundation" composed of ontologies would make the agents more amenable to work together. Reasoning on ontological models has proved valuable in supporting many context-aware applications. Yet, it is far from being the panacea for achieving context-awareness. Moore et al. [29] pointed out that ontologies fails to address the issues of context processing. Roussaki et al. [30] further elaborated on these issues.

4.3.3 Shortcomings with "Modeling" Context

Arguments have recently been aired that the current notion of "context" and hence context-aware computing builds on a positivist philosophical stance, where "context" is stable, delineable, and sense-able information separated from human activity. The argument is that the notion of "context" as referring to the "usage context" for a specific person using some technology cannot be separated from the human activity. "Context" then becomes firmly tied to "meaning" i.e., that context cannot be seen (and much less sensed) as an objective entity in the world, but only exists in connection to subjective meaning in an activity [43].

4.4 Conclusion

Research on context-awareness has been conducted over a decade, but modeling and processing contextual information continues to pose some major challenges. The intrinsic complexity of representing such heterogeneous and volatile information adds to a large range of applications. Although each of the various proposals have addressed reasonable requirements for ubiquitous computing systems, it is unclear whether they can be generalized to suit all context-aware applications. Indeed there is no well-defined standard metric to actually evaluate the advantages and disadvantages of contextual models.

The systems that tend to be completely general and support a wide range of context models and applications often fail to be effective. In fact, the practical applicability and usability are important parameters to determine the quality of context modeling, and they are often inversely proportional to the generality of the model: the more expressive and powerful, the less practical and usable. Different context subproblems and applications have almost incompatible requirements, and there is no common and standard solution. That is the reason why the context models are defined on a per-application basis.

Part II

Contribution

In some areas all the steps are equally elegant and interesting, whereas in other areas we see elegant steps with a lot of dull work in between, usually suppressed by the author. In such areas people are likely to hate formal verification.

– N.G. de Bruijn

5

Formal Query Language

Contents

THE TWENTY-FIVE-YEAR-OLD VISION of a ubiquitous computing world inspired by Mark Weiser [2] has spread out of research laboratories, and has become a tangible reality in our everyday lives [44]. While connected, people do not sit still behind desktop computers any longer, but move within their daily environments (e.g. homes, offices, campuses, cities), using more and more sophisticated devices, such as cell phones and wearable computers. Thus, they continuously modify the set of –potentially mobile– services and objects they may come to interact with. That is a reason why location-awareness has become the most popular feature of ubicomp systems. It enables emerging applications to sense, and adapt to, locations of miscellaneous entities, making the experience and interaction of users with intelligent surroundings easier and more appealing. We believe that location-dependent query processing is a major requirement of ubiquitous computing systems, being closely related to how we model and organize location information.

5.1 Querying Location Models

Location modeling is an extensively studied topic within the ubicomp community. Many models were investigated, along with philosophical discussions about the concept of location per se. Previous works on location modeling focused on the needs of particular applications such as smart rooms [45] and navigation systems [46], or relied on specific tracking technologies, e.g. the Global Positioning System (GPS) [47], radio-frequency identification (RFID) [48]. As far as we know, no model was used as an explicit data structure for location-dependent query processing.

Figure 5.1 illustrates a typical location-aware system and points out the crucial role played by the location model. Updated by the tracking system, the location model represents the physical world, identifies and locates users/objects, and provides an interface for various services.

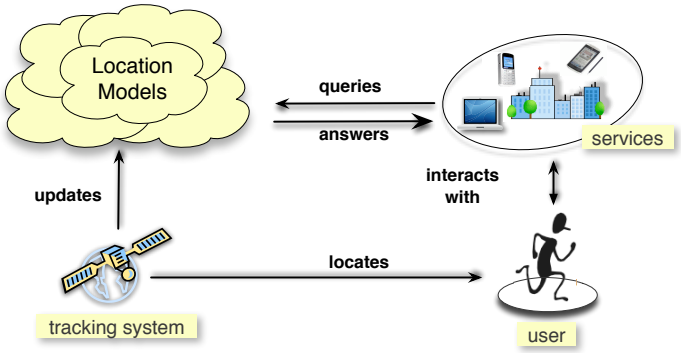

Figure 5.1: Location-aware system.

5.1.1 Location Query Processing

Location query processing corresponds to mobile search over (fragments of) location models, in which the actual service issuing the queries has its counterpart representation. As described by Becker et al. [10], location-dependent queries encompass position queries ("where is the conference room?"), nearest neighbor queries ("where is the closest sushi restaurant to my house?"), navigation queries ("how can I reach the bus terminal?"), and range queries ("what are the convenience stores located in my neighborhood?"). Although the sources and targets of queries can be mobile, our approach does not assume continuous queries.

5.1.2 Motivation and Approach

A general framework for location query processing is necessary so that users, objects and services can effectively benefit from their location-aware surroundings. We aim at providing a query language for location-aware services by advocating the need for defining theoretical foundations. Indeed, modern database systems and standard query

languages (e.g. SQL) rely on first-order logic, efficiently implemented using relational algebra [49].

Our approach rests on a symbolic model of space, along with the hierarchical containment relation between places. We extend this commonly used representation to a semantic model for hybrid logics [50], and thus map location query processing into a model checking framework.

Thereafter, we exhibit the connection between query processing and model checking (Sect. 5.2). We then formally extend the hierarchical space tree to a data structure that suits the requirements of model checking (Sect. 5.3). We then define the formal syntax and semantics of the query language (Sect. 5.4), and outline the implementation of our location query processing framework (Sect. 5.5). After outlining related works (Sect. 5.6), we conclude and discuss future works (Sect. 5.7).

5.2 Background

5.2.1 Towards Symbolic Location Models

Leonhardt [51] proposed a taxonomy for location models and pointed out two major categories: *geometric* and *symbolic* models. Geometric models only represent positions of entities as coordinates. For example, the most prominent reference coordinate system for outdoor environments is the GPS, widely used for navigation applications. However, such low-level information has little utility for human interaction and lacks semantics to describe relations between locations. In existing applications, coordinates are therefore contextualized with auxiliary human-readable information used in city maps. Several proposals [52, 7] deal with higher-level information represented by symbolic models, the notion of *place* which has been defined as:

> "[...] a human-readable labeling of positions. A more rigorous definition is an evolving set of both communal and personal labels for potentially overlapping geometric volumes. An object contained in a volume is reported to be in that place", Hightower [53].

Location information is then defined by the interrelations between all the places and

is commonly represented by a hierarchical space tree (Fig. 5.2). The nodes represent places, and the edges represent containment relations between these places. We propose to derive location query processing into a dynamic graph search, and advocate a model checking approach to formally address this problem.

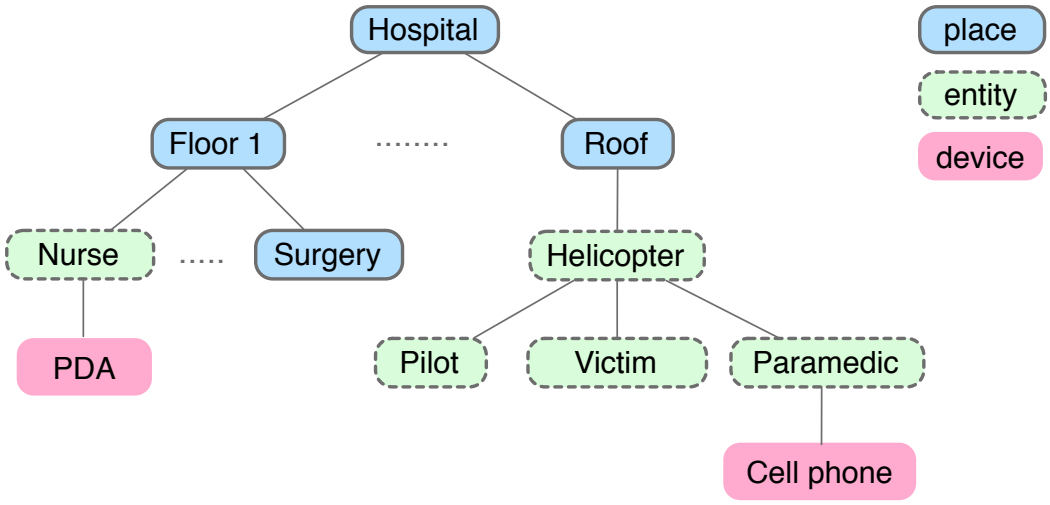

Figure 5.2: Example of hierarchical space tree for a hospital.

5.2.2 Query Processing and Temporal Model Checking

Model checking [54] is a well-founded and widely used approach to analyze various computing systems, both hardware and software. Model checking aims at automatically checking if a given system S satisfies its specification P. For example, S can be an automatic lighting system in a conference room, whose main specification P is to switch on when at least one person is inside. Formally, inputs to a model checking algorithm are:

- a finite transition graph called *Kripke structure* [55]. A Kripke structure is a directed graph whose nodes represent the reachable states of the system and whose edges represent state transitions,

- a property written in a suitable temporal logic formula.

The connection between model checking and query processing has been studied in the literature, because both evaluate logic-based formulas over finite data structures. In particular, researchers have investigated model checking approaches to query semi-structured data based on temporal logic such as the Computational Tree Logic (CTL) [55]. However, temporal logic does not suit location query processing, because of at least two major limitations:

- *Unnamed states*: As mentioned by Franceschet et al. [56], temporal logic lacks mechanisms to name individual nodes of the Kripke structure, whereas location-related queries deal with identified places. Moreover, temporal logics cannot express relative node names that are useful when the user has only a limited knowledge of his environment. Figure 5.3 illustrates such a relative reference: the user queries a place close his current location "here",

- *Top-to-bottom query routing*: Because temporal logic defines operators for current and future states, location queries can only be routed downward the containment hierarchy. Due to (1) the hierarchical nature of the location information and (2) the heterogeneity of sources and targets of queries, our language should provide both downward and upward operators to navigate within the space tree.

Since temporal logic does not meet our requirements, we assume from now on *hybrid logics* [50].

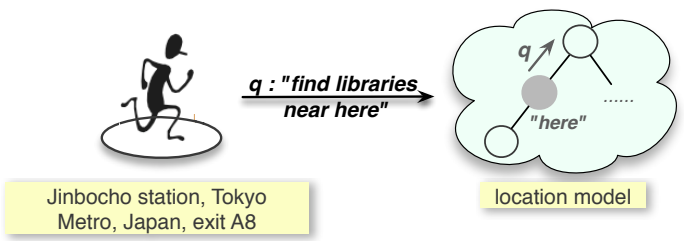

FIGURE 5.3: Example of a query with a relative location.

5.2.3 Hybrid Logics

Hybrid logics introduce the concept of *nominals*. Nominals are propositional variables that are true at exactly one node in the Kripke structure, i.e. if p is a nominal, the formula p holds if and only if the current node is called p. Thus, it is easy to capture the notion of place: a place is associated to a nominal. Besides, hybrid logics define the following operators:

- access operator $@_p$: It gives random access to the place named p. The formula $@_p\phi$ holds if and only if ϕ holds at the place p,
- downarrow binder $\downarrow x$: It creates a new name x and assigns it to the current node. The formula $\downarrow x.\phi$ holds if and only if ϕ holds whenever the current node has been named x.

5.3 Hierarchical space graph: a semantic and data model

In order to implement a hybrid logics-based strategy for querying locations over symbolic location models, we extend the hierarchical space tree to a Kripke-like structure that serves as both a semantic and data model, as follows:

Definition 1 (Hierarchical space graph) *A Hierarchical space graph is a 4-tuple* $G = \{N, R_{\downarrow}, R_{\uparrow}, L : N \rightarrow \mathcal{P}(N)\}$ *in which:*

- *Places are represented by the set* N *of nodes,*
- *Transitions* $R_{\downarrow} \subseteq N \times N$ *and* $R_{\uparrow} \subseteq N \times N$ *define the containment relation between parents and children,*
- *Places are labeled with their children's names by the function* $L : N \rightarrow \mathcal{P}(N)$*,* $\mathcal{P}(N)$ *being the powerset of* N*.*

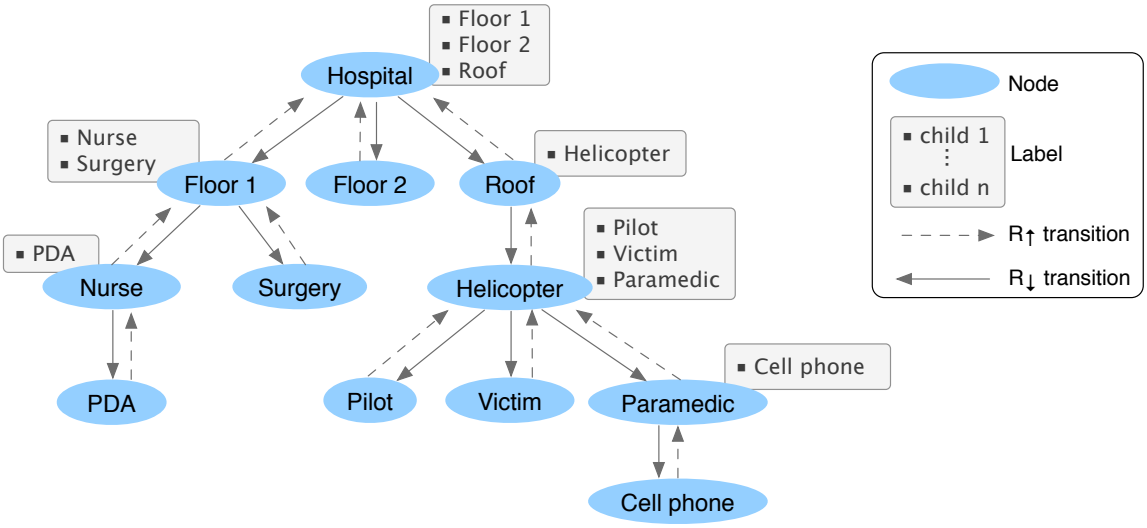

FIGURE 5.4: Hierarchical space graph extending space tree of Fig. 5.2.

The hierarchical space graph of Fig. 5.4 extends the space tree of Fig. 5.2. Labels are Boolean variables that refer to the places contained under a given location. For example, the helicopter returning from a rescue operation may either be on the roof of the hospital or still on its way back. In the former case, the place *Roof* would be labeled by its child *Helicopter*, whereas in the latter case it would have no label *Helicopter*, as the place *Helicopter* would not be its child.

Definition 2 (Transition relations) *The transitions $R_\downarrow$ and $R_\uparrow$ are defined as:*

1. *$R_\downarrow$ and $R_\uparrow$ are irreflexive, intransitive, asymmetric,*

2. $\forall\ p_1, p_2 \in N : (p_1, p_2) \in R_\downarrow \Leftrightarrow (p_2, p_1) \in R_\uparrow$,

3. $\forall\ p_1, p_2, p_3 \in S : (p_1, p_2), \in R_\uparrow\ \wedge\ (p_3, p_2) \in R_\downarrow \Rightarrow p_1 = p_3$.

The property (1) ensures that a place cannot be its own parent or own child (irreflexivity). The intransitivity and asymmetry are trivial. The property (2) defines the relation between the parent p_1 and its child p_2. Finally (3), a place cannot have multiple parents, i.e. there is no overlapping area.

5.4 Hybrid Logics-Based Query Language

In the previous section, we defined a semantic model to express hybrid logic-based queries on the location data. "Where is the nurse?" consists of exploring the hierarchical space graph and finding the place labeled by *Nurse*.

5.4.1 Syntax

Definition 3 (Language components) *The query language $\mathcal{Q}$ contains:*

- *a countable set $N = \{p_1, p_2, \ldots, p_n\}$ of places,*
- *a countable set X of variables $x_1, x_2, \ldots, x_n$, used for bound names,*
- *standard logical symbols $\vee$, $\wedge$,*
- *spatial modalities $E_\downarrow$ and $E_\uparrow$,*
- *hybrid logics operators $\downarrow x$ (binder) and $@_p$ (random access)*

The first point we want to draw to the readers attention is that we identify the places in the model. That is semantically the most important ingredient of hybrid logics. Given these identifiers (e.g. *Hospital*, *Roof*):

- $@_l$ gives direct access to the location l,
- $\downarrow x$ creates a brand new name (or label) x and assigns it to the current location. For example, if the current location is *Roof*, the formula $\downarrow here$ instanciates the name *here* and binds it to *Roof*. As we mentioned earlier, the binding operator allows us to express relative locations.
- $E_\downarrow$ is the spatial counterpart of the CTL temporal operator EF [54], and $E_\uparrow$ its backward analogue. $E_\downarrow$ and $E_\uparrow$ are routing triggers that move queries along the space graph, to both directions.

Definition 4 (Assignment) *An assignment b for G is a mapping $b : X \to N$. Given an assignment b, a variable $x \in X$, and a place $p \in N$, we define b_p^x by setting $b_p^x(x) = p$.*

Every bound name is stored by the assignment b. For example, the binder $\downarrow Emergency$ triggered at location $Helicopter$ creates the name $Emergency$ as an alias for $Helicopter$. The assignment b provides a lookup function to retrieve aliases, e.g. $b_p^x(Emergency) = Helicopter$.

Definition 5 (Query language) *The well-formed formulas of our language are given by the following recursive grammar:*

$$FORM ::= true \mid p \mid \neg FORM \mid FORM \wedge FORM$$

$$\mid E_{\uparrow}FORM \mid E_{\downarrow}FORM \mid @_pFORM \mid \downarrow x.FORM$$

in which $p \in N$ *and* $x \in X$.

5.4.2 Semantics

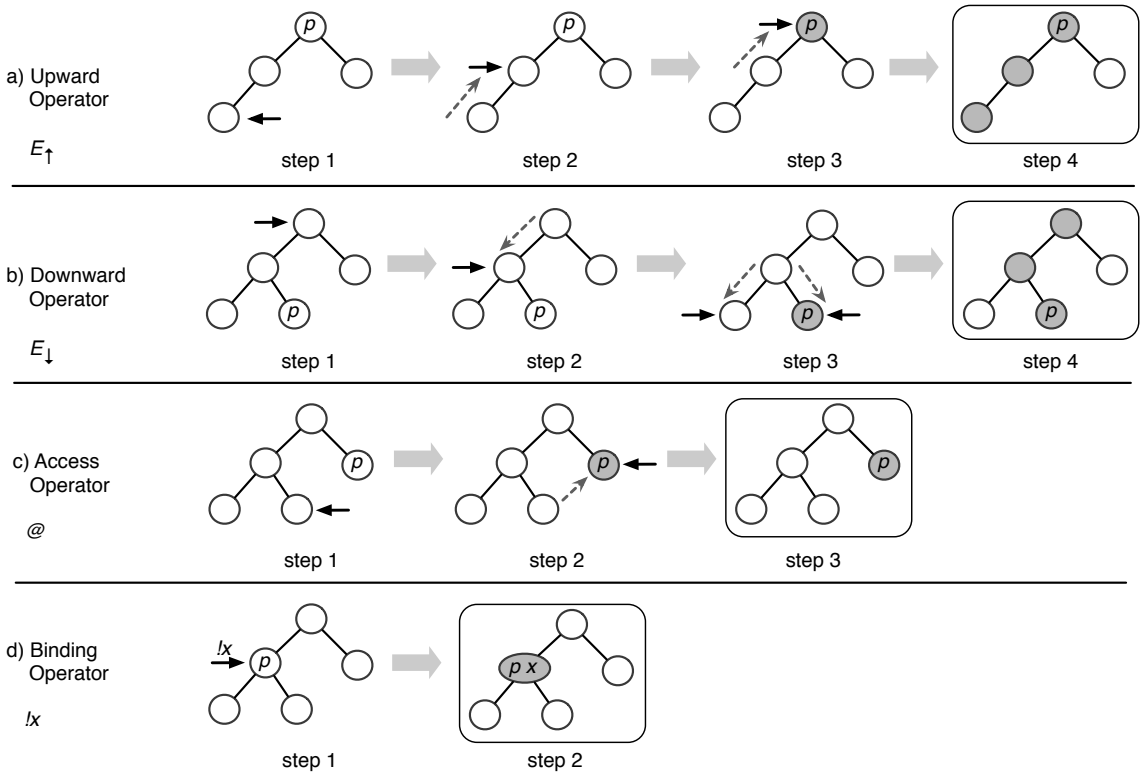

FIGURE 5.5: The four hybrid operators in action.

The semantics of the hybrid logic formulas, i.e. their formal interpretation in terms of hierarchical space graphs, is defined. The exact meaning of a formula is given by

a satisfaction relation connecting the hierarchical space graph with a formula, and is written as: $G, b, l \models \varphi$.

Readers who are not familiar with such a notation should regard the left side of the relation as the context of the query, and the right side as the expression of the query.

- *Context*: It encompasses the hierarchical space graph G representing the location model (i.e. the data structure), the binding names locally created by the query (and stored by the assignment b) and the location l, where the query is being evaluated.

- *Query*: It is expressed by a hybrid logics-based formula. We give below its formal semantics, some informal comments, as well as illustrations to demonstrate the expressiveness of our query language.

1. $G, b, l \models true$

   True is valid everywhere.

2. $G, b, l \models p \Leftrightarrow p$ is true at l

   Place p is contained in location l. For example, the relation $G, b, floor1 \models nurse$ is satisfied, i.e. the nurse is on the first floor.

3. $G, b, l \models \neg\varphi \Leftrightarrow G, b, l \not\models \varphi$

   This is the standard definition of $\neg$ operator. For example, the relation $G, b, surgery \models \neg victim$ is satisfied, i.e. the victim is not in the surgery.

4. $G, b, l \models \varphi_1 \wedge \varphi_2 \Leftrightarrow (G, b, l \models \varphi_1) \wedge (G, b, l \models \varphi_2)$

   This is the standard definition of $\wedge$ operator. For example, the relation $G, b, helicopter \models victim \wedge paramedic$ is satisfied, i.e. both the victim and the paramedic are inside the helicopter.

5. $G, b, l \models \varphi_1 \vee \varphi_2 \Leftrightarrow (G, b, l \models \varphi_1) \vee (G, b, l \models \varphi_2)$

   This is the standard definition of $\vee$ operator. For example, the relation $G, b, helicopter \models nurse \vee paramedic$ is satisfied, i.e. either the nurse or the paramedic is inside the helicopter.

6. $G, b, l \models E_{\uparrow}\varphi \Leftrightarrow \exists l' \in N : l'$ is reachable from l following $R_{\uparrow}$ transitions and $(G, b, l' \models \varphi)$

   See Fig. 5.5(a). Operator $E_{\uparrow}$ routes the query upward, searching for a place that satisfies formula φ (steps 1 and 2). Place p is found (step 3), so all the nodes between p and the source of the query also satisfy φ (step 4).

7. $G, b, l \models E_{\downarrow}\varphi \Leftrightarrow \exists l' \in N$, such that l' is reachable from l following $R_{\downarrow}$ transitions and $(G, b, l' \models \varphi)$

   See Fig. 5.5(b). Operator $E_{\downarrow}$ routes the query downward, searching for a place that satisfies formula φ (steps 1 and 2). Place p is found (step 3), so all the nodes between p and the source of the query also satisfy φ (step 4). For example, the relation $G, b, \mathit{hospital} \models E_{\downarrow}\mathit{surgery}$ is satisfied, i.e. the surgery room is located on the first floor.

8. $G, b, l \models @_p\varphi \Leftrightarrow \big((G, b, p \models \varphi \text{ for } p \in N) \text{ or } (G, b, g(p) \models \varphi \text{ for } p \in X)\big)$

   See Fig. 5.5(c). Operator $@_p$ straightly forwards the query to the specific place p (step 1), where it is evaluated (step 2). Name p can either be (1) a place identifier or (2) a bound name. In the latter case, a name resolution is first performed. For example, the relation $G, b, \mathit{hospital} \models @_{\mathit{roof}}\mathit{helicopter}$ is satisfied, i.e. the helicopter is on the roof.

9. $G, b, p \models \downarrow x.\varphi \Leftrightarrow G, b^x_p, p \models \varphi$

   See Fig. 5.5(d). Operator $\downarrow x$ binds the variable x to the place p, where the query is evaluated (step 1). Within the same query, x can then be used as an alias for p (step 2). For example, the formula $\big(G, b, \mathit{hospital} \models (E_{\downarrow} \downarrow \mathit{emergency}.\mathit{victim}) \wedge (@_{\mathit{emergency}}\mathit{paramedic})\big)$ is satisfied, i.e. there is an emergency place in the hospital where the paramedic is assisting the victim. *Emergency* is used as an alias for $Helicopter$.

5.5 Implementation

We built an implementation of the query processing framework, named *Chequery* which architecture is depicted in Fig. 5.6. The language itself is independent of any programming languages but the current implementation uses *OCAML* (version 3.09 or later).

Chequery works in interactive mode: users can change the source of the query by moving along hierarchical space graphs[1] and execute location queries based on the syntax specified in Sect. 4.

5.5.1 Architecture

Tree-Based Database

The first version of our system supports filesystem-based tree structures, that are mapped to hierarchical space trees. Files are associated with places, and hierarchical links are associated with containment relations between places.

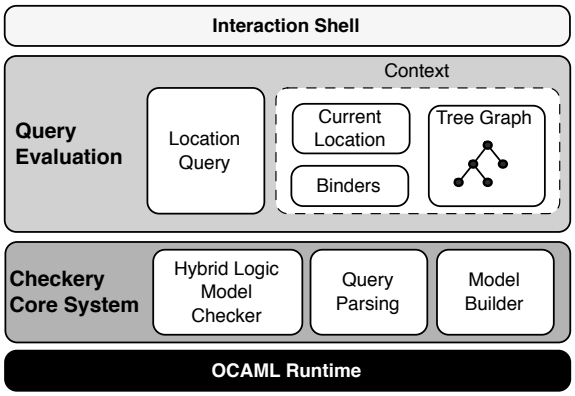

Figure 5.6: System architecture.

Hierarchical Space Graph

Model Builder processes the input files and extracts their spatial organisation to produce a corresponding modal structure, i.e. a hierarchical space graph which:

[1] This navigation system may be related to the navigation inside tree-based filesystems (i.e. *enter* and *leave* a folder).

1. Associates files with places,

2. Links the places with transition relations according to the containment structure,

3. Labels each place with its contained places. Thus, nodes of the hierarchical tree graph are labeled places, called *worlds*[2].

```
type world =
    {mutable container: place ;
     mutable valid_prop: place list;
     mutable nested_loc: place list;}
```

The data (or Kripke) structure has the form *((place world (worlds))...)*. It is implemented in a hashtable, whose keys are world labels and values are worlds. Note that in the current implementation, after *Model Builder* is executed, functions implemented in *Hybrid Logic Model Checker* use the same modal structure for all queries.

5.5.2 Query Processing

The recursive grammar defining our query language is syntactically analysed by the *Query Parsing* module. Query evaluations call the *Hybrid Logic Model Checker*, that maintains and updates two kinds of data, the context and the current query expression. Without any occurrence of binders, queries simply returns Boolean values. Let us consider the following query (see Fig. 5.4):

$$G, b, hospital \models E_{\downarrow} nurse$$

Query evaluation is actually a matching procedure between the hierarchical space graph G and the logical expression $E_{\downarrow} nurse$. This relation can be expressed by "is the nurse inside the hospital?". The answer would be either "yes" (true) or "no" (false). Presence is a low-level information, captured by e.g. a sensing system using RFID tags. Location-based services are either searching for objects or people location. Binder

[2] "World" is the appropriate term in model checking terminology.

operator $\downarrow x$ enriches the relations, so that results of the model checking algorithm are either failures or bound places where formulas are satisfied. For example:

$$G, b, hospital \models E_{\downarrow}(\downarrow place.nurse)$$

The result is the nurse's current location bound to *place*. The meaning of the relation changes from "is there a place inside the hospital in which the nurse is located?" to "where is the nurse?". The module *Binders* maintains the bound variables in a list of pairs {*bound variable, real place*}. The names of *Real places* are resolved after a computation, carried out by a sub-routine of the query evaluator.

5.6 Related Works

Existing research on location-awareness focused on the design and implementation of application-specific services (e.g. [45, 46]). As a result, data management for location information attracted little attention. Many location-based services maintain and process location information in ad-hoc manners or rely on centralized and inadequate data infrastructures. As far as we know, the underlying connection between location modeling and location query processing has not been explored yet.

The DOMINO project [57] indirectly enables this connection by modeling moving objects, and their dynamic locations, in a database. The authors focus on trajectories of moving objects and propose a set of spatial and temporal operators to query locations. However, the data model is built on a centralized architecture and cannot represent any relation between objects and places. Still in the context of moving objects databases, Güting and Schneider [58] propose a modal logic-based approach for query processing. They define a Future Temporal Logic (FTL) to reason about future events[3] using SQL-like query language extensions. However, drawbacks include (1) the location data are stored in a single data repository, (2) the approach is dedicated to outdoors environments, along with GPS (i.e. a geometric location model), and (3) trajectories of objects must be known beforehand by the query processor.

[3]The authors consider estimations about trajectories of objects.

5.7 Conclusion

We presented a novel approach to handle location query processing in pervasive computing environments. Starting from symbolic location models (hierarchical space trees), we defined an extended Kripke-like structure, as both a semantic and data model for a hybrid logic-based query language. The logic is defined by four modal operators and provides good expressiveness for common location queries. We implemented a prototype system based on the framework. General, our approach is general is not be limited to location query processing. Actually, it has a broad application domain and can be applied to any hierarchical data structure.

In some areas all the steps are equally elegant and interesting, whereas in other areas we see elegant steps with a lot of dull work in between, usually suppressed by the author. In such areas people are likely to hate formal verification.

– N.G. de Bruijn

6 Application

Contents

CHALLENGES for implementing large-scale context-aware systems abound. Among these are issues related to the representation, storage, and processing of contextual information in terms of data structures and algorithms [59]. This chapter discusses these issues within the scope of location-aware services. We argued in the previous chapters that data management for context-aware services has been overlooked thus far. We now describe our approach and present its underlying principles, which revolve around model checking and logic-based graph querying.

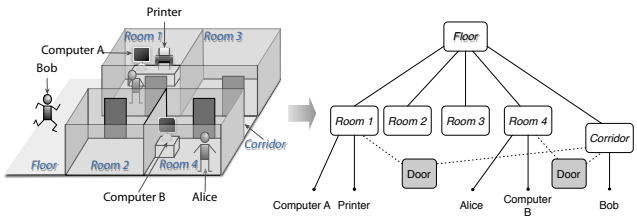

FIGURE 6.1: Real-world space and its corresponding graph-based data structure.

6.1 Query Language

From the stance of information management, contextual information is data that has to be captured, processed, and managed by the system. Context modeling shares very similar characteristics with data modeling in the database community. As Angles and Gutiérrez pointed out, the relational model is designed for simple record-type data – where the schema is fixed – and the query language cannot explore the underlying graph of the relationship among the data such as paths, neighborhoods, and patterns.

We need a query language which mechanisms can directly refer to the graph structure corresponding to contextual information in the real world. Associated with graphs

are specific graph operations in the query-language algebra, such as finding shortest paths and determining certain subgraphs. Explicit graphs and graph operations allow users to express a query at a high level of abstraction. To some extent, this is the opposite of graph manipulation in deductive databases, where often, fairly complex rules need to be written. It is not important to acquire full knowledge of the structure to express meaningful queries.

Like SQL, our query language needs a theoretical foundation behind it, because it must be sound in the sense that the results of a query (i) do not miss any information that can satisfy the necessary and sufficient conditions of the query and (ii) do not contain any information that cannot satisfy the conditions.

6.1.1 Query Processing

Pervasive-computing environments have several unique requirements, which existing location-based or personalized-information services in non-pervasive computing environments do not. These are as follows:

Scalability

Pervasive computing infrastructures need to maintain and process a vast amount of heterogeneous, dynamic information. Indeed, the trend is toward the deployment of large-scale systems, e.g. city-wide systems, which maintain information about the location of moving objects and adaptive services. Query mechanisms require scalability and response. Scalability not only encompasses the increase in information maintained by the system but also the number of queries that need to be processed.

Absence of Relational Database

Query processing can be implemented by general purpose object-oriented database systems. However, these systems tend to have a large footprint. The majority of devices that make up pervasive-computing environments only have limited capabilities in terms of bandwidth, autonomy and computational power. The data models, and

location models in particular, that are maintained by these devices need to cope with these limitations.

Remote Access and Control

Moreover, query processing should not be limited to data retrieval. Indeed, as a pervasive-computing environment consists of service-provider devices, e.g., remotely-controllable appliances, query processing should also provide access and control to remote devices and services.

6.2 Approach

This section details our approach to addressing the problems of querying symbolic location models in context-aware systems.

6.2.1 Model Checking

Model checking [54, 60] is a well-founded approach and has been a widely used technique to automatically analyze various types of computing systems, ranging from hardware to software. A system is said to be verified if it satisfies the specifications defined by its designers. A specification is called a *property*. For example, an automatic lighting system in a conference room could have the following property: to switch on when at least one person is present in the room.

A model-checking algorithm takes in input (i) the representation of the system S to be analyzed and (ii) the property P. In more formal terms, S is defined as a finite state machine usually called a Kripke structure. A Kripke structure is no more than a directed graph with nodes representing the possible states of the system and edges represent the transitions (or triggers) between states. In our previous example, someone entering or leaving the room could be a trigger. The property P is written in a suitable temporal logic formula.

The connection between model checking and query processing has been extensively

studied in the literature, as both evaluate logic-based formulas over finite data structures.

6.2.2 Application to Context-aware Computing

The various algorithms implemented in model checkers cannot be directly applied to querying information in data-intensive, dynamic environments. Indeed, while model checking does offer a mathematically sound framework for matching graph-like structures, it is nevertheless a static, off-line process. Therefore, it needs to be adjusted to cope with the requirements of context-aware services.

Moreover, temporal logic lacks any mechanism for naming individual nodes in the Kripke structure. Indeed, during the verification phase the graph is parsed from its unique entry node (which corresponds to the initial state of the system). Location-related queries need such mechanisms to handle identified places. Moreover, temporal logic cannot express relative nodes that are required when users only have local knowledge about the system.

Because temporal logic defines the operators for current and future states, location queries can only be routed down the location model. Due to innate decentralized information on location and the various query sources and targets, our query language should some flexible operators to search the space graph.

In our previous paper [5], we proposed the use of model checking principles as query mechanisms for context-aware computing. The paper aimed at defining the theoretical aspects of model-checking for context-aware computing. The formal semantics of the modal language presented in Table 6.1. Queries are defined by the logical assertion $G, b, l \models \varphi$, where G is the graph-based data structure (i.e. the Kripke structure corresponding to the location model), b acts as a container for virtual nodes created on-the-fly (i.e. *bound* nodes), l the location where the query is evaluated, and φ the logic-based formula (i.e. the actual query).

On the other hand, this paper addresses the design and implementation of a logic-based query language and its model checking-based processing for context-aware computing. Furthermore, the framework presented in this paper has many features that

TABLE 6.1: Formal semantics of the query language

$G, b, l \models true$		
$G, b, l \models p$	iff	p is true at l
$G, b, l \models \neg\varphi$	iff	$G, b, l \not\models \varphi$
$G, b, l \models \varphi_1 \wedge \varphi_2$	iff	$(G, b, l \models \varphi_1) \wedge (G, b, l \models \varphi_2)$
$G, b, l \models \varphi_1 \vee \varphi_2$	iff	$(G, b, l \models \varphi_1) \vee (G, b, l \models \varphi_2)$
$G, b, l \models E_{\uparrow}\varphi$	iff	$\exists l'(lRl' \wedge (G, b, l' \models \varphi)$
$G, b, l \models E_{\downarrow}\varphi$	iff	$\exists l'(lR^{-}l' \wedge (G, b, l' \models \varphi)$
$G, b, l \models @_p\varphi$	iff	$G, b, p \models \varphi$
$G, b, p \models\downarrow x.\varphi$	iff	$G, b^x_p, p \models \varphi$

its earlier version did not have, although the former is defined and constructed based on the theoretical foundation of the latter. For example, the language itself has been extended to satisfy the requirements of query processing for context-aware computing and provide existing location models with query functions.

6.2.3 Logic-based Graph Querying

As mentioned before, the earlier version of the query language was only defined formally, i.e. queries were expressed by satisfaction relations. Since SQL-based "select-from" syntax is by far the most well-known and most widely used, we chose to embed our logic-based query language in a more readable SQL-like language. From now on, we process queries in an SQL-fashion.

All the queries that we consider allow expressions for navigating the data graph. Some offer the possibility of comparing object identities, which allows us to implement queries with joins. We will focus on monadic queries only, i.e., queries that return a set of objects of the database, or, equivalently, a set of nodes of the graph structure. The reason for this restriction is that we want to embed the query-processing problem into the global model-checking problem, as the output of a global model checker is a set of nodes. In the following the examples are taken from Fig. 6.1.

Simple Selection – Query Q_1

`select` X

`from` *path* X

`where` X.*condition*

Intuitively, Q_1 retrieves all nodes reachable through *path* satisfying the filter *condition*. The variable X is used as a container for these nodes. For example, the following query selects all the available `printers` located in `room_1`:

```
select X
from floor.room_1 X
where X.(type::printer and status::idle)
```

Multiple-path Selection – Query Q_2

`select` X_i

`from` $path_1$ X_1, $path_2$ X_2, ... , $path_n$ X_n

`where` $condition_1, \cdots, condition_n$

Intuitively, the schema Q_2 binds the variable X_1 to the nodes reachable through $path_1$, it binds the variable X_2 to the nodes reachable through $path_2$, and so forth. The expression X.*condition* filters the nodes placed in the variable X according to the Boolean filter *condition*. Finally, the nodes contained in the focus X are selected. For example, the following query selects the status of all `printers` in `Room_4`:

```
select Y
from room_4 X, X.status Y
where X.type::printer
```

Advanced Selection – Query Q_3

`select` X_i

`from` $sexp_1$, $sexp_2$, ... , $sexp_n$

`where` $condition_1$, $condition_2$, ..., $condition_m$

Each $sexp_j$ is a selection expression and each $condition_j$ is a filter expression. A selection expression is either *path* X or $X.path\ Y$, where X and Y are variables. A filter expression is either $X.condition$, or $X = Y$, or $X = Y$, where X and Y are variables and *condition* is a filter expression as in Q. From example, the following query selects rooms with at least two persons:

```
select X
from floor X, X.person Y, X.person Z
where Y <> Z
```

Ontologies

We could use ontology-based models for describing context as complementary symbolic location models. Indee, ontology-based models help in defining the vocabulary of use, as well as inferring higher-level contexts, while the structure of the location information encapsulates the semantical and topological relationships within the environment. The two can be combined to describe both the vocabulary and the structure of the environment, thus enhancing the expressivity of the queries.

6.3 Implementation

Our framework is implemented using Objective-C. Although the approach is independent of any programming language, we chose Objective-C because (i) it compiles native code and (ii) provides garbage collection.

6.3.1 System Architecture

`LocationModel` maintains the graph-based data structure that represents the information about the physical entities of the environment. `LocationModel` inputs the *abstract* data representation of the underlying sensing devices, and build the corresponding modal structure, i.e., the Kripke structure which:

1. Associates nodes with object entities.

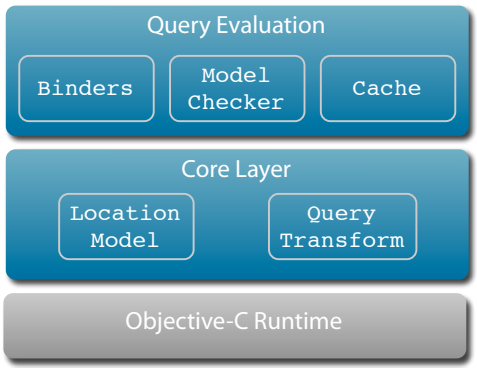

Figure 6.2: Layered architecture.

2. Links the entities with their corresponding links.

3. Initialize the object entities with their various attributes.

6.3.2 Model Checking Algorithm

`ModelChecker` receives:

- A hybrid model $M = M, R, V$.
- An assignment g.
- A hybrid formula ϕ (see Table 6.1).

After termination, every node in the model is labeled with the sub-formulas of ϕ that hold at that state, i.e. the reachable nodes. Our algorithm uses a bottom-up strategy; it examines the sub-formulas of ϕ in increasing order of length, until ϕ itself has been checked.

Let R be an accessibility relation; then R^- is the inverse of R: Rvu if, and only if, Ruv. For $n \geq 1$, let $R^n(w)$ be the set of states that are reachable from w in n R^- steps, and $R^{-n}(w)$ be the set of states that are reachable from w in n R^--steps. The states belonging to $R^1(w)$ are successors of w, while those belonging to $R^1(w)$ are predecessors of w. Given a model $M = M, R, V$, we denote by M the model M, R, V. The length of a formula ϕ, denoted by $|\phi|$, is the number of operators (Boolean and

modal) of ϕ plus the number of atoms (propositions, nominals, and variables) of ϕ. Let $sub(\phi)$ be the set of sub-formulas of ϕ.

`ModelChecker` updates a table L of size $|\phi| \times |M|$ whose elements are bits. Initially, $L(\alpha, w) = 1$ if, and only if, α is an atomic proposition in $sub(\phi)$ such that $w \in V(\alpha)$. When the evaluation terminates, $L(\alpha, w) = 1$ if, and only if $M, g, w \models$ for every $\alpha \in sub(\phi)$. Given $\alpha \in sub(\phi)$ and $w \in M$, we denote by $L(\alpha)$ the set of states $v \in M$ such that $L(\alpha, v) = 1$ and by $L(w)$ the set of formulas $\beta \in sub(\phi)$ such that $L(\beta, w) = 1$. The engine uses subroutines `MCF`, `MCA`, and `MC@` to check the respective sub-formulas of the form $F\alpha$, $A\alpha$, and $@\alpha$. The pseudocode for these procedures is presented in Figure 6.3.

6.3.3 Query Processing

Queries are syntactically analysed and translated into their logic-based versions by the `QueryTransform` module. Query evaluations call the `ModelChecker`, which maintains and updates the context and the current expression of the query. Without any occurrence of binders, queries simply returns Boolean values:

$$G, b, floor \models E_{\downarrow} Alice$$

The query evaluation is a matching procedure between the graph-based data structure G and the logical expression $E_{\downarrow} Alice$. This relation can be expressed by "is Alice inside the room?". The answer would be either "yes" (true) or "no" (false). Binder operator $\downarrow x$ (i.e., `select` X) enriches the relations, so that `ModelChecker` returns the locations where the given formulas are satisfied. For example, consider the query:

$$G, b, floor \models E_{\downarrow}(\downarrow place.Alice)$$

The result is Alice's current location bound to *place*. The meaning of the relation changes from "is there a place in which Alice is located?" to "where is Alice?". `Binders` maintains the bound variables in a list of pairs {`bound variable, location`}.

```
1: procedure MCF(M, g, α)
2:     for w ∈ L(α) do
3:         for v ∈ R^{-1}(w) do
4:             L(Fα, w) ← 1
5:         end for
6:     end for
7: end procedure
8: procedure MCA(M, g, α)
9:     if L(α) = M then
10:        for v ∈ M do
11:            L(Aα, w) ← 1
12:        end for
13:    end if
14: end procedure
15: procedure MC@(M, g, t, α)
16: Let w = [V, g](t)
17:    if L(α, w) = 1 then
18:        for v ∈ M do
19:            L(@_t α, v) ← 1
20:        end for
21:    end if
22: end procedure
```

FIGURE 6.3: Pseudocode of the subprocedures `MCF`, `MCA`, and `MC@` called by the query processing engine.

6.4 Application

In this section, we present the contribution of our query language from a software engineering perspective. We have a project to provide a location/user-aware system for several museums, that assist visitors. Although the current version has no query mechanism, a prototype implementation of the next generation of the system is based

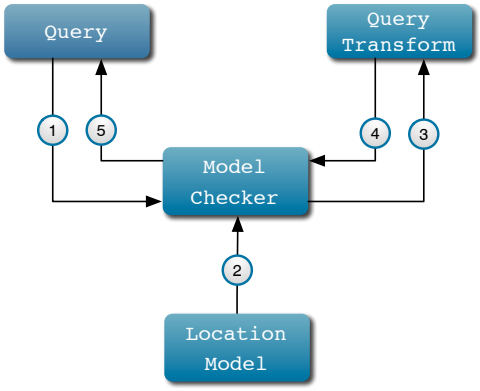

FIGURE 6.4: Query workflow.

on the query language presented in the paper, we discuss how our query language can enhance the overall system.

6.4.1 Application to RFID-based Systems

Although the proposed query processing framework is general and may be adjusted to cope with many different location-based services (e.g. through language extensions), this section highlights its application for indoor environments, along with RFID-based tracking system. Due to the symbolic location model, the query language obviously deals with qualitative information about the location tree structure, rather than quantitative information (e.g. distances between physical entities). From our point of view, RFID-based tracking systems are a convenient application domain because they inherently handle the symbolic notion of *place*, through RFID readers' coverage areas.

Such systems detect the location of physical entities, and usually deploy services bound to the entities at proper computing devices near where they are located. A pair of RFID-reader and RFID-tag thus define the containment relation presented in this paper, the former being the parent, and the latter, the child. When a RFID-tag is detected inside a RFID-reader's coverage area, the underlying hierarchical tree graph is updated: the tag becomes the reader's child, and labels its parent (see Fig. 6.5). However, the language could be applied to geometric-based systems, where a prior refinement step derives symbolic relations from geometric coordinates.

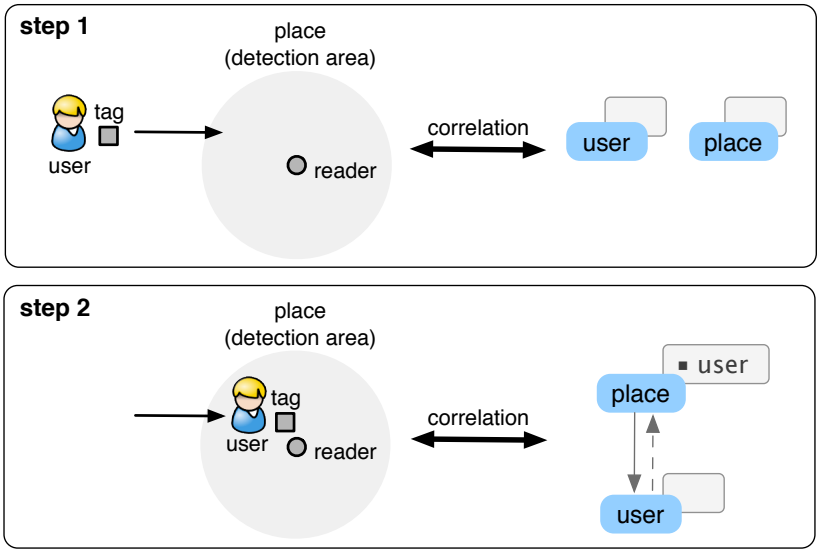

Figure 6.5: Correlation between RFID-based tracking system and symbolic location model.

6.4.2 Location-aware System for Museums

The system is deployed in the National Museum of Nature and Science in Ueno (Japan). It was constructed with the framework presented in [7]. This system:

- Plays audio-annotations at specified spots when visitors stand at the spots,
- Provides compound document-based GUIs for selecting audio-annotations according to visitors and spots. These GUIs visualize the positions of visitor.

Each visitor is provided with a hat equipped embedding active RFID-tags to track their locations. The exhibition space is augmented with RFID-tag readers that detect the presence of a visitor within it. When visitors come sufficiently close to some objects (e.g. fossils of dinosaur), located at several spots in the exhibition, the system selects and plays sounds about dinosaurs according to the combination of the spot, the visitor and his/her route in the room. The upper right picture (2) of Fig. 6.6 depicts the equipment used during the experiment: a loud speaker and a RFID reader positioned in front of fossils.

The query processing engine is independent of any hardware and computing device. Within the current system in the museum, the devices are not overloaded with query

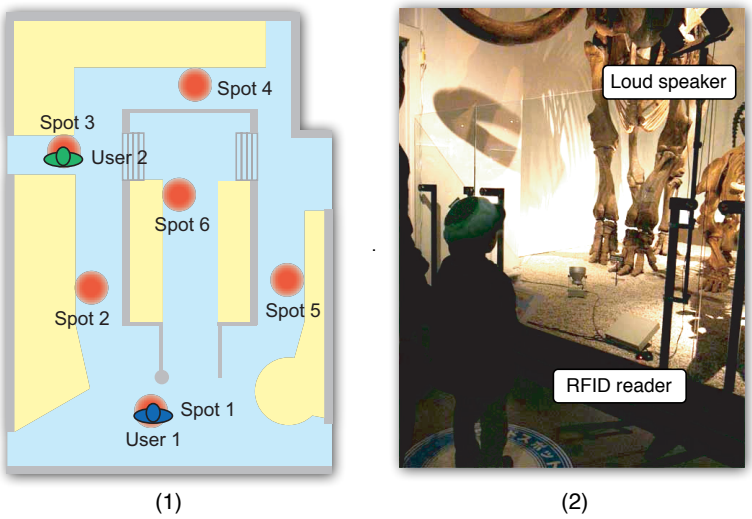

FIGURE 6.6: (1) Map-based GUI and (2) RFID speaker.

processing tasks, as query processing is performed on proxies where the application is deployed. Our solution doesn't make any assumption about the deployment of the application, that goes beyond the scope of this paper.

6.4.3 Query Language for Middleware Support

Even if the application is fully functional, its future extensions are made difficult from an engineering point of view. Without any query mechanisms, the current location-based service (i.e. the appropriate audio-based information) is delivered to the users (i.e. the visitors) in an reactive manner, i.e. the detection that a user enters a spot triggers the right loud speaker. This event-based mechanism is dependent on the underlying sensing systems and it is specifically implemented for that application-specific. Instead, our query processing framework could be integrated as an available component at the middleware level. In the current version of the system, developers need to *manually* construct the query mechanisms, based on simple events, e.g. entering or leaving an exhibit spot. Our query processing framework is useful when we want to extend the functionalities of such a location-based system, without increasing its complexity.

Activity Monitoring

Collecting information about the activities of visitors is an on-going process that is relevant for museum curators. The information delivered by monitoring is analyzed and used to improve and enhance the collections. In our location-based service, the query processing framework could be used to determine visitors' itineraries inside the museums. For example, visitors may not visit all the exhibits, or visit some of them more than one time during his/her course. Our query language can then enhance the existing information system of the museum, by providing accesses to location information.

Event-based Action Triggers

The reactive location-based service, based on simple location triggers, points out the lack of a query language. Let us consider the following example. When a visitor enters a spot S at time t_0, the appropriate speaker delivers some sound-based information for a given amount of time T, i.e. the event trigger is pulled. If a second visitor enters the same spot S at time $t_1 < t_0 + T$, nothing happens because the event triggers has not been released yet. The information that a spot is already visited can be accessed by simple location queries using our framework. Our language provides a higher level of control for the management of event triggers, and eases the development of more complex behaviors. In the presented system, this management is done in a centralized way, by a software component that has the knowledge of the whole location model. Thus, the spot is dedicated to only deliver location information. Our query language could be used by the spots themselves to get and access location-based information from a more local view, and then triggers given actions.

Performance Evaluation

Although the system has not been tested in a real environment yet, we made a simulation of three scenarios, each of them corresponding to a given number of visitors inside the exhibition rooms of the museum (i.e. number of places in the hierarchical tree graph). We generated 20 queries of different size. Queries were processed locally,

on a Pentium CoreDuo 1.5GHz laptop.

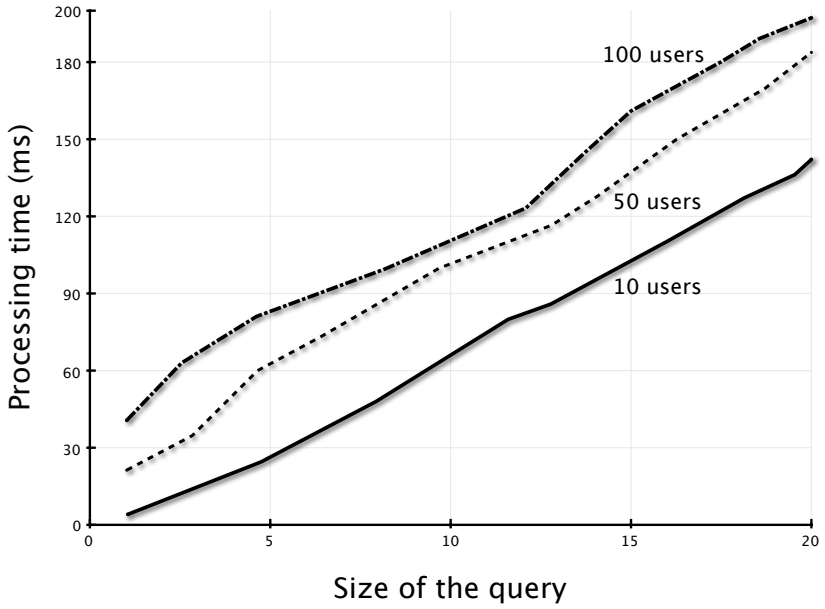

FIGURE 6.7: Time to process 20 queries of different size with respectively 10, 50, and 100 users.

The current system was not optimized for performance; however, the cost for evalutating and processing a query is shorter than 200 ms[1]. We believe that this cost is acceptable for a location-aware system used in rooms or buildings.

A word on complexity: Franceschet and de Rijke [61] investigated the model-checking problem for hybrid logics. They obtained lower bounds on the computational complexity of the model checking problem for hybrid logics with binders. They proved that the addition of nominals and the @ operator did not increase the complexity of the model checking task. In contrast, whenever hybrid binders are present in the language, the running time of the resulting model checker is exponential in the nesting level of the binders. Finally, they proved that the model-checking problem for hybrid logics with binders is PSPACE-complete.

Note that our approach does not suffer from the so-called state-explosion problem, which can occur when the state graph is computed. Since we rely on sensors to acquire

[1]In a real environment, this evaluation time may be seriously affected by many different parameters (e.g. network latency).

location data from the environment, we do not algorithmically need to construct such a state graph. The location model is merely an input for the query-processing framework.

6.5 Related Work

Since our approach combines the application of formal methods to context-aware computing on one side, and query-processing mechanisms on the other, we have presented the work related to both aspects in this section.

Ilarri et al. [39] investigated query processing for location-based services. They targeted outdoor applications that have to handle a fair amount of moving objects in real-time. Responsiveness and performance are therefore critical. The authors focused on geometric location models that provide both quantitative and high-resolution data on moving objects. Acknowledging that fine-grain GPS coordinates may be inappropriate for many location-based applications (high-resolution requires more overhead), they introduced the notion of location granules. Granules are geographical areas that can be defined at different scales, e.g. freeways and buildings. However, the semantics of the granules are rather poor, since relationships between granules are not explicitly defined. Therefore, the query language used to retrieve the location cannot benefit from the spatial semantics of the granules.

Aiming at providing contextual-information retrieval that is both scalable and robust, Roussaki et al. [30] enhanced the database with some location-based retrieval mechanisms. They implemented a hybrid context model that maintained (i) symbolic entities like streets and buildings, and (ii) objects located in a coordinate system. The context database maintained a hierarchy of entities. By combining the semantical expressiveness of ontologies with the hierarchical structure of tree-based location models, they could dispatch and route the queries along the context database hierarchy and obtain a significant increase in information retrieval. The same approach has been developed further to improve context filtering [32]. However, query processing was achieved in an ad-hoc manner and the query language was not explicitly defined.

Grossniklaus [40] proposed an object-oriented version control to address the challenges of data management in context-aware services. The query-processing framework was application-independent. Indeed, as many frameworks are specific to a single application domain, their notion of context (in particular the context dimensions e.g. users, devices, and environmental factors) is only relevant to a subset of applications. Moreover, the representation and storage are often implied by the context model. Their approach is general and is not limited to any particular context model. It is similar to Bolchini et al.'s [13] which emphasizes the importance of context-based data tailoring.

Ranganathan and Campbell [62] proposed a formal approach to describing the properties and capabilities of pervasive computing environments. The underlying description language was based upon the ambient calculus which describes the entities located in the environment, i.e., the *ambients*, in a tree-based topology of boundaries. The ambitious goal is provide a framework that empower the designers to prove the properties of pervasive computing environments. The properties expressed in ambient logic can encompass mobility, security and interactions between ambients. Such endeavors can be related to the work of Franceschet and de Rijke [61] who proposed a combined model-checking framework that they applied to the specification and verification of various properties of mobile systems. Complex spaces would require a model checker.

Perich et al. [41] proposed a solution for managing data in pervasive-computing applications. It broadly consisted of two parts: treating the devices as semi-autonomous entities guided in their interactions by profiles and context, and designing a contract-based transaction model. The profile was grounded in a semantically rich language for representing information about users, devices, and data objects each described in terms of "beliefs", "desires", and "intentions" - a model which has been explored in multi-agent interactions. They introduce data-based routing algorithms and semantic-based data caching and replication algorithms enabling mobile devices to utilize their data-intensive vicinities.

In some areas all the steps are equally elegant and interesting, whereas in other areas we see elegant steps with a lot of dull work in between, usually suppressed by the author. In such areas people are likely to hate formal verification.

– N.G. de Bruijn

7

Conclusion

We presented a query-processing framework for location-based services and demonstrated its applicability to existing symbolic location models. The proposed approach aimed at complementing existing work wherein the location information is modeled in a symbolic fashion independently of any sensing technology. Indeed, current location models lack any mechanism for querying the location of people, objects, or services, or do so in an ad-hoc fashion. We have shown that formal methods can be used as the underlying query-processing mechanisms. Indeed, we have built our query language from model-checking principles. This well-founded language can query graph-based data structures. We can therefore guarantee that the results of any query (i) do not miss any information that can satisfy the necessary and sufficient conditions of the query and (ii) do not contain any information that cannot satisfy the condition.

We would like to emphasize that our approach is not limited to location-aware

services. The approach we presented is independent of any sensor technologies as it relies on an abstraction of location information. Also, the proposed query-processing engine can be implemented in existing middleware. Our contribution is to complement existing location models such as hierarchical- or graph-based models. We believe that our system can be easily applied to context-aware services, because the approach itself can support general contextual information that can be modeled in a graph-structure.

References

[1] T. Strang and C. Linnhoff-Popien. *A context modeling survey.* In *Proceedings of 1st International Workshop on Advanced Context Modelling, Reasoning and Management, in coordination with the 6th International Conference on Ubiquitous Computing (UbiComp 2004)*, Lecture Notes in Computer Science (Springer, 2004).

[2] M. Weiser. *The computer for the twenty-first century.* Scientific American pp. 94–100 (1991).

[3] A. Silberschatz, H. F. Korth, and S. Sudarshan. *Data models.* ACM Computing Surveys **28**(1), 105 (1996).

[4] R. Angles and C. Gutiérrez. *Survey of graph database models.* ACM Computing Survey **40**(1), 1 (2008).

[5] C. Hoareau and I. Satoh. *Query language for location-based services: A model checking approach.* IEICE Transactions on Information and Systems **E91**(D4), 976 (2008).

[6] Y.-B. Kang and Y. Pisan. *A survey of major challenges and future directions for next generation pervasive computing.* In *Proceedings of the 21th International Symposium on Computer and Information Sciences (ISCIS 2006)*, Lecture Notes in Computer Science, pp. 755–764 (Springer, 2006).

[7] I. Satoh. *A location model for pervasive computing environments.* In *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)*, pp. 215–224 (IEEE, 2005).

[8] M. Strimpakou, I. Roussaki, C. Pils, M. Angermann, P. Robertson, and M. E. Anagnostou. *Context modelling and management in ambient-aware pervasive environments.* In *Proceedings of the 1st International Workshop on Location- and Context-Awareness (LoCA 2005)*, Lecture Notes in Computer Science, pp. 2–15 (Springer, 2005).

[9] M. Baldauf, S. Dustdar, and F. Rosenberg. *A survey on context-aware systems.* The International Journal of Ad Hoc and Ubiquitous Computing **2**(4), 263 (2007).

[10] C. Becker and F. Dürr. *On location models for ubiquitous computing.* Personal and Ubiquitous Computing **9**(1), 20 (2005).

[11] C. Jiang and P. Steenkiste. *A hybrid location model with a computable location identifier for ubiquitous computing.* In *Proceedings of the 4th International Conference on Ubiquitous Computing (UbiComp 2002)*, Lecture Notes in Computer Science, pp. 307–313 (Springer, 2002).

[12] A. K. Dey. *Understanding and using context.* Personal and Ubiquitous Computing **5**(1), 4 (2001).

[13] C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. *A data-oriented survey of context models.* SIGMOD Record **36**(4), 19 (2007).

[14] A. Schmidt, M. Beigl, and H.-W. Gellersen. *There is more to context than location.* Computers & Graphics **23**(6), 893 (1999).

[15] D. Salber, A. K. Dey, and G. D. Abowd. *The context toolkit: aiding the development of context-enabled applications.* In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '99)*, pp. 434–441 (ACM, New York, NY, USA, 1999).

[16] J. E. Bardram. *The java context awareness framework (jcaf) - a service infrastructure and programming framework for context-aware applications.* In *Proceedings of the 3rd International Conference on Pervasive Computing (Pervasive 2005)*, Lecture Notes in Computer Science, pp. 98–115 (Springer, 2005).

[17] J. Indulska, T. McFadden, M. Kind, and K. Henricksen. *Scalable location management for context-aware systems.* In *Proceedings of the 4th IFIP WG6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2003)*, Lecture Notes in Computer Science, pp. 224–235 (Springer, 2003).

[18] G. Biegel and V. Cahill. *A framework for developing mobile, context-aware applications.* In *Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*, pp. 361–365 (IEEE, 2004).

[19] G. Chen and D. Kotz. *A survey of context-aware mobile computing research.* Tech. rep., Dartmouth College (2000).

[20] B. Schilit, N. Adams, and R. Want. *Context-aware computing applications.* In *Proceedings of the Workshop on Mobile Computing Systems and Applications*, pp. 85–90 (IEEE, 1994).

[21] J. Indulska, R. Robinson, A. Rakotonirainy, and K. Henricksen. *Experiences in using cc/pp in context-aware systems.* In *Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003)*, Lecture Notes in Computer Science, pp. 247–261 (Springer, 2003).

[22] A. Ranganathan, R. E. McGrath, R. H. Campbell, and M. D. Mickunas. *Use of ontologies in a pervasive computing environment.* The Knowledge Engineering Review **18**(3), 209 (2003).

[23] E. Katsiri and A. Mycroft. *Applying bayesian networks to sensor-driven systems.* In *Proceedings of the 10th IEEE International Symposium on Wearable Computers (ISWC'03)*, pp. 149–150 (IEEE, 2003).

[24] K. Henricksen, J. Indulska, and A. Rakotonirainy. *Modeling context information in pervasive computing systems.* In *Proceedings of the 1st International Conference on Pervasive Computing (Pervasive 2002)*, Lecture Notes in Computer Science, pp. 167–180 (Springer, 2002).

[25] T. Sohn. *Context-aware computing support for the educationally disadvantaged.* In *Proceedings of the IEEE Symposium on Human Centric Computing Languages and Environments (HCC 2003)*, pp. 293–294 (IEEE, 2003).

[26] H. Chen, T. Finin, and A. Joshi. *An ontology for context-aware pervasive computing environments.* Knowledge Engineering Review **18**(3), 197 (2003).

[27] S. W. Loke. *Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective.* The Knowledge Engineering Review **19**(3), 213 (2004).

[28] P. Korpipää and J. Mäntyjärvi. *An ontology for mobile device sensor-based context awareness.* In *Proceedings of the 4th International and Interdisciplinary Conference Modeling and Using Context (CONTEXT 2003)*, Lecture Notes in Computer Science, pp. 451–458 (Springer, 2003).

[29] P. Moore, B. Hu, and J. Wan. *Smart-context: A context ontology for pervasive mobile computing.* The Computer Journal pp. 1–17 (first published on March 4, 2007, doi:10.1093/comjnl/bxm104).

[30] I. Roussaki, M. Strimpakou, N. Kalatzis, M. Anagnostou, and C. Pils. *Hybrid context modeling: A location-based scheme using ontologies.* In *Proceedings of the 4th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW '06)*, pp. 2–7 (IEEE, 2006).

[31] N. Noy. *Order from chaos.* Queue **3**(8), 42 (2005).

[32] C. Pils, I. Roussaki, and M. Strimpakou. *Location-based context retrieval and filtering.* In *Proceedings of the 2nd International Workshop on Location- and Context-Awareness (LoCA 2006)*, Lecture Notes in Computer Science, pp. 256–273 (Springer, 2006).

[33] D. R. McCarthy and U. Dayal. *The architecture of an active data base management system.* In *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, pp. 215–224 (ACM, 1989).

[34] J. Widom and S. Ceri. *Introduction to active database systems.* In *Active Database Systems: Triggers and Rules For Advanced Database Processing*, pp. 1–41 (Morgan Kaufmann, 1996).

[35] I. Satoh. *A location model for smart environments.* Pervasive and Mobile Computing **3**(2), 158 (2007).

[36] E. F. Codd. *A relational model of data for large shared data banks.* Communications of the ACM **13**(6), 377 (1970).

[37] R. H. Güting. *An introduction to spatial database systems.* The International Journal on Very Large Data Bases (VLDB) **3**(4), 357 (1994).

[38] M. Levene and G. Loizou. *The nested relation type model: An application of domain theory to databases.* Computer Journal **33**(1), 19 (1990).

[39] S. Ilarri, E. Mena, and C. Bobed. *Processing location-dependent queries with location granules.* In *Proceedings of the 2nd OnTheMove Workshop on Pervasive Systems (PerSys'07)*, Lecture Notes in Computer Science, pp. 856–865 (Springer, 2007).

[40] M. Grossniklaus. *Context-Aware Data Management. An object-Oriented Version Model* (Verlag Dr. Müller, 2007).

[41] F. Perich, A. Joshi, T. Finin, and Y. Yesha. *On data management in pervasive computing environments.* IEEE Transactions on Knowledge and Data Engineering **16**(5), 621 (2004).

[42] N. Davies and H.-W. Gellersen. *Beyond prototypes: Challenges in deploying ubiquitous systems.* IEEE Pervasive Computing **1**(1), 26 (2002).

[43] P. Dourish. *What we talk about when we talk about context.* Personal and Ubiquitous Computing **8**(1), 19 (2004).

[44] G. Bell and P. Dourish. *Yesterday's tomorrows: Notes on ubiquitous computing's dominant vision.* Personal and Ubiquitous Computing **11**(2), 133 (2007).

[45] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. *Easyliving: Technologies for intelligent environments.* In Springer, ed., *Proceedings of the 2nd International Symposium on Handheld and Ubiquitous Computing (HUC'00)*, pp. 12–29 (2000).

[46] C. Stahl and D. Heckmann. *Using semantic web technology for ubiquitous hybrid location modelling.* In *Proceedings of 1st Workshop on Ubiquitous GIS, in conjunction with 12th International Conference on Geoinformatics* (2004).

[47] N. Marmasse and C. Schmandt. *A user-centered location model.* Personal and Ubiquitous Computing **6**(5-6), 318 (2002).

[48] T. Tsukiyama. *Global navigation system with rfid tags.* In *Proceedings of the SPIE International Society for Optical Engineering*, pp. 256–264 (2001).

[49] M. Negri, G. Pelagatti, and L. Sbattella. *Formal semantics of sql queries.* ACM Transactions on Database Systems (TODS) **16**(3), 513 (1991).

[50] C. Areces and B. ten Cate. *Handbook of Modal Logic*, chap. Hybrid Logics (Elsevier, 2005).

[51] U. Leonhardt. *Supporting Location-Awareness in Open Distributed Systems*. Ph.D. thesis, Department of Computing, Imperial College, London, UK (1998).

[52] M. Beigl, T. Zimmer, and C. Decker. *A location model for communicating and processing of context*. Personal and Ubiquitous Computing **6**(5-6), 341 (2002).

[53] J. Hightower. *From position to place*. In *Proceedings of the Workshop on Location-Aware Computing*, pp. 10–12 (2003).

[54] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking* (MIT Press, 2000).

[55] E. M. Clarke and ernd Holger Schlingloff. *Handbook of automated reasoning*, chap. Model Checking, pp. 1635–1790 (Elsevier, 2001).

[56] M. Franceschet, A. Montanari, and M. D. Rijke. *Model checking for combined logics with an application to mobile systems*. Automated Software Engineering **11**(3), 289 (2004).

[57] O. Wolfson, S. Chamberlain, K. Kalpakis, and Y. Yesha. *Modeling moving objects for location based services*. In *Proceedings of the US NSF Workshop on Developing an Infrastructure for Mobile and Wireless Systems*, pp. 46–58 (Springer, 2001).

[58] R. H. Güting and M. Schneider. *Moving Objects Databases* (Morgan Kaufmann Publishers Inc., 2005).

[59] M. Satyanarayanan. *Pervasive computing: Vision and challenges*. IEEE Personal Communications **8**(4), 10 (2001).

[60] C. Baier and J.-P. Katoen. *Principles of Model Checking* (MIT Press, 2008).

[61] M. Franceschet and M. D. Rijke. *Model checking hybrid logics (with an application to semistructured data)*. Journal of Applied Logic **4**(3), 279 (2006).

[62] A. Ranganathan and R. H. Campbell. *Provably correct pervasive computing environments*. In *Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communications (PerCom 2008)*, pp. 160–169 (IEEE, 2008).

[63] T. Kindberg, J. J. Barton, J. Morgan, G. Becker, D. Caswell, P. Debaty, G. Gopal, M. Frid, V. Krishnan, H. Morris, J. Schettino, B. Serra, and M. Spasojevic. *People, places, things: Web presence for the real world*. MONET **7**(5), 365 (2002).

[64] A. Oulasvirta. *When users "do" the ubicomp*. interactions **15**(2), 6 (2008).

[65] C. K. Hess and R. H. Campbell. *A context-aware data management system for ubiquitous computing application*. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS 2003)* (IEEE, 2003).

[66] K. Henricksen and J. Indulska. *A software engineering framework for context-aware pervasive computing.* In *Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*, pp. 77–86 (IEEE, 2004).

[67] D. Nicklas and B. Mitschang. *On building location aware applications using an open platform based on the nexus augmented world model.* Software and System Modeling **3**(4), 303 (2004).

[68] S. Duval and C. Hoareau. *Fundamental needs in intelligent environments: Specificities for older adults.* In *Workshops Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (WAINA 2008)*, pp. 850–855 (IEEE, 2008).

[69] C. Hoareau and I. Satoh. *Hybrid logics and model checking: A recipe for query processing in location-aware environments.* In *Proceedings of the 22nd International Conference on Advanced Information Networking (AINA 2008)*, pp. 130–137 (IEEE, 2008).

[70] S. Duval and C. Hoareau. *Design of a ubiquitous system for affective bonding and support within the family.* In *Workshops Proceedings of the 22nd International Conference on Advanced Information Networking and Applications (WAINA 2008)*, pp. 844–849 (IEEE, 2008).

[71] C. Hoareau and I. Satoh. *A model checking-based approach for location query processing in pervasive computing environments.* In *Proceedings of the Second OTM International Workshop on Pervasive Systems (PerSys 2007)*, Lecture Notes in Computer Science, pp. 866–875 (Springer, 2007).

[72] J. Ye, L. Coyle, S. Dobson, and P. Nixon. *A unified semantics space model.* In *Proceedings of the 3rd IEEE International Symposium on Location- and Context-Awareness (LoCa 2007)*, Lecture Notes in Computer Science, pp. 103–120 (Springer, 2007).

[73] M. Kaenampornpan and E. O'Neill. *An intergrated context model: Bringing activity to context.* In *Proceedings of the 1st International Workshop on Advanced Context Modelling, Reasoning and Management, in coordination with the 6th International Conference on Ubiquitous Computing (UbiComp 2004)*, Lecture Notes in Computer Science (Springer, 2004).

[74] Y. Cao, R. Klamma, M. Hou, and M. Jarke. *Follow me, follow you - spatiotemporal community context modeling and adaptation for mobile information systems.* In *Proceedings of the 9th International Conference on Mobile Data Management (MDM 2008)*, pp. 108–115 (IEEE, 2008).

[75] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. *Experiences of developing and deploying a context-aware tourist guide: the guide project.* In *Proceedings of the*

6th International Conference on Mobile Computing and Networking (MobiCom 2000), pp. 20–31 (ACM, 2000).

[76] R. Hull and R. King. *Semantic database modeling: Survey, applications, and research issues.* ACM Computing Survey **19**(3), 201 (1987).

[77] S. Peters and H. E. Shrobe. *Using semantic networks for knowledge representation in an intelligent environment.* In *Proceedings of the 1st IEEE International Conference on Pervasive Computing and Communications (PerCom'03)*, pp. 323–329 (IEEE, 2003).

[78] S. A. Dobson and P. Nixon. *More principled design of pervasive computing systems.* In *Proceedings of the 9th IFIP Working Conference on Engineering for Human-Computer Interaction (EHCI'04)*, Lecture Notes in Computer Science, pp. 292–305 (Springer, 2004).

[79] T. Chaari, D. Ejigu, F. Laforest, and V.-M. Scuturici. *A comprehensive approach to model and use context for adapting applications in pervasive environments.* Journal of Systems and Software **80**(12), 1973 (2007).

[80] D. Raptis, N. K. Tselios, and N. M. Avouris. *Context-based design of mobile applications for museums: a survey of existing practices.* In *Proceedings of the 7th Conference on Human-Computer Interaction with Mobile Devices and Services (Mobile HCI 2005)*, pp. 153–160 (ACM, 2005).

[81] A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster. *The anatomy of a context-aware application.* Wireless Networks **8**(2-3), 187 (2002).

[82] J. Kjeldskov and M. B. Skov. *Exploring context-awareness for ubiquitous computing in the healthcare domain.* Personal and Ubiquitous Computing **11**(7), 549 (2007).

[83] J. Pascoe, N. Ryan, and D. Morse. *Issues in developing context-aware computing.* In *Proceedings of the 1st International Symposium on Handheld and Ubiquitous Computing (HUC'99)*, pp. 208–221 (Springer, 1999).

[84] A. Held, S. Buchholz, and A. Schill. *Modeling of context information for pervasive computing applications.* In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)* (2002).

[85] O. Lehmann, M. Bauer, C. Becker, and D. Nicklas. *From home to world - supporting context-aware applications through world models.* In *Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communications (PerCom 2004)*, pp. 297–308 (IEEE, 2004).

[86] P. J. Brown, J. D. Bovey, and X. Chen. *Context-aware applications: from the laboratory to the marketplace.* IEEE Personal Communications **4**(5), 58 (1997).

[87] G. D. Abowd, C. G. Atkeson, J. Hong, S. Long, R. Kooper, and M. Pinkerton. *Cyberguide: a mobile context-aware tour guide.* Wireless Networks **3**(5), 421 (1997).

[88] K. Henricksen, J. Indulska, and A. Rakotonirainy. *Using context and preferences to implement self-adapting pervasive computing applications.* Software: Practice and Experience **36**(11-12), 1307 (2006).

[89] G. Chen, M. Li, and D. Kotz. *Data-centric middleware for context-aware pervasive computing.* Pervasive and Mobile Computing **4**(2), 216 (2008).

[90] J. E. Bardram, T. R. Hansen, M. Mogensen, and M. Søgaard. *Experiences from real-world deployment of context-aware technologies in a hospital environment.* In *Proceedings of the 8th International Conference on Ubiquitous Computing (UbiComp 2006)*, Lecture Notes in Computer Science, pp. 369–386 (Springer, 2006).

[91] M. A. Munoz, M. Rodríguez, J. Favela, A. I. Martinez-Garcia, and V. M. González. *Context-aware mobile communication in hospitals.* Computer **36**(9), 38 (2003).

[92] A. Zimmermann, M. Specht, and A. Lorenz. *Personalization and context management.* User Modeling and User-Adapted Interaction **15**(3-4), 275 (2005).

[93] J. Coutaz, J. L. Crowley, S. Dobson, and D. Garlan. *Context is key.* Communications of the ACM **48**(3), 49 (2005).

[94] J. Ye, L. Coyle, S. Dobson, and P. Nixon. *Ontology-based models in pervasive computing systems.* The Knowledge Engineering Review **22**(4), 315 (2007).