2
Fundamental Concepts

In this chapter we present the main concepts underlying in this thesis. In the first section, we describe our motivating scenario, to which we will refer along this work, in order to exemplify and clarify our approach. In Section 2.2, we discuss commonly found characteristics of Ambient Intelligence. In Section 2.3 we discuss context-awareness, a fundamental aspect in AmI systems, and in 2.4 the use of ontologies as the basis of our context model. In Section 2.5 we discuss context reasoning in general and in Section 2.6 we focus in the rule-based reasoning approach. Finally, in the last section we argue about the interrelation among these topics.

2.1
Scenario

As a typical scenario to exemplify our approach, we consider a fictitious conference on Ubiquitous Computing (UbiConference) where several researchers from different universities and companies gather to present and discuss their recent work. We assume that the conference is divided in several technical sessions on subjects such as *Middleware*, *Ambient Intelligence*, etc, and panel sessions on detached subjects (e.g. *Privacy*). It also comprises workshops on specific subjects such as *Context Modeling and Reasoning*.

Professor Silva is a lecturer and researcher affiliated with the Informatics Department of PUC-Rio. He is also participating in the UbiConference in different roles: (a) he is a member of the Programme Committee (PC); (b) he will chair the *Middleware* session; (c) he will present a paper in the *Context Modeling and Reasoning* workshop; and, of course, (d) he will also be a general attendee of others sessions in the event.

Let us assume that UbiConference takes place in a convention center with several meeting rooms equipped with some infrastructure to support the organizing committee and the attendees with ubiquitous services. A service called Conference Organizer (ConfOrg) is part of this infrastructure and aims at providing context-aware functionalities, such as notifying the participants about the beginning of presentations in which they may be interested, or alerting a PC member when the session chair is absent at the moment it is about to start, for instance.

Previously, when registering at the event website, Silva downloaded and installed in his notebook the Conference Companion (ConfComp), an application provided by the organizers that would help him not only with his agenda during the event, but also with identifying people with interests similar to his, thus stimulating the collaboration and social interactions with other researchers at the event. Let us further assume that ConfComp interacts with ConfOrg to provide the ubiquitous services mentioned before, tailored for Silva's preferences. For that sake, after installing ConfComp, Silva is asked to provide detailed information about his affiliation and subjects of interest. Besides that, Silva's notebook is configured to run an indoor positioning service (e.g., the MoCA's Location Inference Service [35]), capable of continuously determining in which room of the conference center he is located. However, Silva did not agree in disclosing any of his personal data — nor his location nor his preferences — to ConfOrg or to the others attendees.

When arriving at the UbiConference venue, the ConfOrg service detects that his notebook is connected to the local wireless network and automatically registers his presence at the conference. By then, Silva's ConfComp receives the updated schedule of the sessions of that day, which he selected to attend. From this moment on, whenever he is outside the room of a session that he wants to attend and it is about to start, ConfComp notifies him to hurry to the corresponding room. At the moments when there is no session of interest for Silva, ConfComp would suggest him to go to some presentation where a great part of the audience shares similar interests as him.

The described scenario comprises a series of applications and ambient services that exemplify just some of the possible uses for Ambient Intelligence technologies. In the following section we discuss this paradigm in further detail.

2.2 Ambient Intelligence

Ambient Intelligence (AmI), i.e., "intelligent" pervasive computing, builds on three recent key technologies [36]: Ubiquitous Computing [37], Ubiquitous Communication [38] and Intelligent User Interfaces [39]. Ubiquitous Computing is the integration of microprocessors into everyday objects like furniture, clothing, white goods, toys, even paint. Ubiquitous Communication enables these objects to communicate with each other and the user by means of *ad hoc* wireless networking. Intelligent User Interfaces enable the inhabitants of an AmI environment to control and interact with the environment in a natural (voice, gesture) and personalized way (preferences, context). In an AmI environment, massively distributed devices operate collectively, while embedded in the environment, using information and "intelligence" that is hold by the interconnected system [40].

AmI aims at making use of those entities in order to provide users with an environment that offers services when and if needed. As such, an AmI system has to be (a) unobtrusive, i.e., its services must not intrude on the user's consciousness unless he needs them; (b) personalized, i.e., it must be able to recognize the user and tailor its behavior to the user's needs; (c) adaptive, i.e., its behavior can change in response to a person's actions and environment's context; and (d) anticipatory, i.e., it must anticipate a person's desires and environment as much as possible without mediation [41].

An example of an environment enriched with AmI is a "smart home", where several domestic artifacts and items can be enriched with sensors to gather information about their use and in some cases even to act independently without human intervention [11]. This approach enables to achieve increased safety, comfort, or economy [42], e.g., by monitoring the activities of the user and providing assistance when a possibly harmful situation is developing, adjusting temperature automatically or turning off lights in an empty room, for instance.

AmI may also help impaired people to live independently, improving their access to a wide range of services and facilities [13]. Automated home care systems based on AmI technology aim at the prolongation of an independent life of assisted persons in their own homes, reducing the dependency on intensive personal care to a minimum and thereby increasing the quality of life for the affected group while substantially decreasing the costs for society [43].

Academic environments are also target area in which AmI systems — or prototypes — has particularly flourished. In places such as universities, research centers, conference rooms, etc., where lecturers and students engage in learning activities, researchers gather to run technical meetings, attendees join to listen to technical presentations, AmI technologies are useful to facilitate the interaction among all participants. A plethora of projects have presented solutions targeting these same spaces, such as Gaia [18], CoBrA [19], CHIL [22], Semantic Space [32], CASMAS [44] and others. This is also the flavor of our scenario, as described in Section 2.1.

One great challenge for AmI environments is how to adequately address the heterogeneity and dynamic nature of users, services and devices [45]. Other key issues in the development of AmI are context-awareness and context-based reasoning and how to identify and provide the most appropriate service for the user and his task [46]. The ultimate goal is to make the ambient services more *intelligent* and adaptive to the specific needs of their users, so that the users do not need to get involved in service discovery, usage and personalization.

Privacy is also a concern in AmI environments, as this technology is regarded as having the potential to create an invisible and comprehensive surveillance network. This is because AmI systems influence two important design parameters relating to privacy: the ability to monitor and the ability to search. Depending on what kind of motives one assumes for preserving privacy, ambient intelligence technology can become the driving factor for changing the scope and impact of privacy protection as it exists today, and creating substantially different social landscapes in the future [47].

Our Assumption. In this thesis, we assume that users carry one or more mobile devices enabled with positioning sensors. They move through different spaces and organizations, and each time they enter a physical environment enriched with AmI technology, applications executing on their devices autonomously interact with different ambient services, i.e., services executing on the ambient infrastructure. These services and applications personalize their functional behavior based on the context data available at the moment, but each has access to different parts of the overall context information (c.f. Section 2.1). Although privacy is a concern in AmI and a motivation to the study of distributed context scenarios, this subject will not be further discussed in this thesis.

2.3 Context-awareness

Context-awareness is the ability of a system to sense the current environment and autonomously perform appropriate adaptations aiming at its optimal operation, customized behavior and facilitated user interaction [9]. When a user changes his context, it is desirable that the applications on his mobile devices be able to adapt to the new situation, and the environment be able to adapt its services according to the presence of the new user.

In AmI environments, more specifically, systems should be aware of the presence of persons in the geographical space, perceive their needs and autonomously make available and personalize services that help users to perform their tasks [21]. In such case, an adaptation may be the triggering of an adequate ambient service tailored for to needs of a user, or having an application running in a user's device starting an action, such as a notification for the user. For example, in our scenario, when a presentation starts a user that is inside a conference room could have his smart-phone's ring turned off, while another user that is chatting at the lobby could be notified about the presentation by the ConfComp application, executing on his notebook.

There exist several definitions for context and context-awareness, but one of the most referenced ones can be found in [26]: "*Any information which can be used to characterize the situation of an entity. An entity is a person, a place or an object which is considered relevant for the interaction between a user and an application, including the user and the application.*" In an attempt to classify context, Chen and Kotz [48] identified four basic types of context: computational context (i.e., state of resources at the device and of the network), user context (i.e., persons, places and objects), physical context (e.g., luminosity, noise, temperature) and temporal context (e.g., time of the day, period of the year, etc.). Abowd et al. [49] proposed the notions of *primary context* (localization, identity, activity and time) and of *secondary context*, where the latter one can be deduced from the former one and may be used for making adaptation decisions at a higher level of abstraction.

Conceptually, context provisioning can be organized in three layers [50]: *data acquisition and distribution*, *interpretation* and *utilization*. The *data acquisition and distribution layer* is responsible for acquiring raw context data from sensors and devices, which need to be interpreted and evaluated with respect to its accuracy, stability and reliability before it can be utilized. The *interpretation layer* is responsible for this operation, and may combine context data from different sources to enhance its reliability or completeness. For that sake, this layer is in charge of performing context reasoning — the focus of this work —, as will be discussed in Section 2.5. The *utilization layer* helps applications to select appropriate actions and adaptations based on the available context information and supports the interactions of the applications with other components of the context-aware system. To reduce the complexity of developing context-aware applications, such systems adopt middleware infrastructures for addressing context provisioning tasks [51].

For applications to be able to select, describe and manage context-aware adaptations — or trigger services and actions —, the applications and the middleware infrastructure have to share a context model. A context model consists in a formal representation used to describe the context information in context-aware systems, so that every piece of context data may be defined, stored and exchanged in a machine processable form [52]. Strang and Linnhoff-Popien [53] identified and compared six types of context models: attribute-value pairs, schema-based models, graphic models, logic-based models, object-oriented models and ontology-based models. The author's main conclusion is that the object-oriented and the ontology-based models are the most complete and expressive ones, and hence are the most suited for modeling context for ubiquitous computing.

Our Assumption. In our scenario, the context information is represented by data available both at the user's devices, such as his location and preferences, and at the ambient infrastructure, such as the list of activities and its status (e.g., a session that is about to start), the room assigned for each activity and the personnel involved. As a means of describing context information involving aspects of the physical environment, computational resources and social aspects, we adopted an ontology-based model, as will be discussed in the next section.

2.4 Ontology-based Context Model

An ontology is a formal, explicit description of the concepts in a particular domain of discourse. It provides a vocabulary for representing domain knowledge and for describing specific situations in a domain. An ontology-based approach for context modeling lets us describe context information semantically and share a common understanding of this information among users, devices and services. The main benefits of this sort of model are (a) enabling the reuse of models, (b) enabling the sharing of common knowledge among several applications [54], and (c) allowing the use of formal analysis of the domain knowledge, such as performing context reasoning to deduce high-level contextual information [55].

Ontologies are semantically rich languages, i.e., have great expression power and means of abstraction. As such, they can express all the relationships, models and diagrams defined by taxonomies [56], relational database schemas [57] or a OO software models [58], as well as n-ary relations, constraints, rules and other differentiators including negation and disjunction [59]. Therefore ontologies have been preferred over other conceptual modeling approaches for representing context information in ubiquitous systems [60].

To ensure effective information sharing among devices, ontologies need to be formal and expressive enough to establish a common terminology that guarantees consistent interpretation [61]. The formalism of choice in ontology-based models of context is typically OWL-DL [62], which is becoming a de facto standard in various application domains and is supported by a number of reasoning services [63]. OWL-DL ontologies map directly to Description Logics (DL) [64], a successful family of logic-based knowledge representation formalisms — consisting in decidable fragments of First Order Logic (FOL) —, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. As such, it has been employed in various application domains, such as natural language processing, configuration, databases or bio-medical ontologies [65], and also in smart spaces ontologies [66].

Differently from FOL, DL explicitly distinguishes between the terminological knowledge (or schema) and the concrete situation [67]. As such, a DL knowledge base — or ontology — consists of two parts: a *terminological part*, the *TBox*, which defines concepts and states additional constraints on the interpretation of these concepts, and an *assertional part*, the *ABox*, which describes individuals and their relationship to each other and to concepts [65]. A TBox comprises (a) *classes*, which represent the concepts of the domain, (b) *properties* that characterize these concepts (*datatype properties*) or define valid relationships between concepts (*object properties*); and (c) *axioms*, which are restrictions applicable to certain elements of the ontology and necessary for a complete description of the knowledge domain. An ABox consists of (a) *individuals* (or instances), defining the concrete elements of a domain associated with every concept in a TBox; and (b) *facts*, or assertions, which associate individuals with specific classes (*unary predicates*) or establish correlations between individuals (*binary predicates*) based on the properties defined in the TBox [68]. In a context-aware system, these facts are used to represent the context data.

For example, in an ontology about the domain of Mobile Devices, the TBox could possibly contain the class *Smart-phone*, which could have as a object property *hasCompany*, indicating the *Company* that builds it, being this another class of the ontology. A possible restriction could state that a *Smart-phone* must have always exactly one *Company* related to it by the property *hasCompany.* As to the ABox, it could contain *SP-1* as an individual of the *Smart-phone* class, defining a specific device in a domain of application, *Nokia* could be an individual of the *Company* class, while *hasCompany(SP-1, Nokia)* would be a fact — a binary predicate in this case — correlating these two individuals, asserting that *SP-1* was manufactured by *Nokia.*

The use of OWL-DL greatly facilitates the modeling of a particular domain of knowledge. Ontologies may be fully described by defining classes, properties, individuals, characteristics of individuals (datatype properties), and relations between individuals (object properties), written manually or with the help of ontology editors, such as Protégé [69]. Complex descriptions of classes and properties can be built by composing elementary descriptions through specific operators provided by OWL-DL [70]. Besides that, an ontology not always has to be entirely described from scratch, as ontologies may be reused or extended to model similar domains. Moreover, OWL-DL ontologies can be verified/classified with the aid of inference mechanisms, e.g., RACER [71] and FaCT [72], for consistency checks, classification and discovery/inference of new information.

In highly dynamic and heterogeneous environments, such as AmI, where different entities may join and leave the environment unforeseeingly, the use of ontologies brings two additional benefits. First, applying ontology integration techniques [73], an ontology that represents a given domain can be dynamically composed from the ontologies that describe the domains of the different interacting elements. In our scenario, for instance, the context of a professor, represented in a *university ontology*, who is inside a room, represented in a *conference center ontology*, carrying a smartphone, represented in a *device ontology*, could be represented by the composition of these three ontologies. Second, as different entities are very likely to employ different knowledge representations, ontology alignment techniques [74] are needed to allow that such representations may be aligned into a single one that can be shared by applications and services.

Our Assumption. To model our AmI system, we took into account not only the physical space (e.g., the modeling of conference rooms, locations) and the availability of resources (e.g., the device's battery level or the quality of network connectivity), but also the social context [75], describing organizational aspects (e.g, sections or departments of a company), users' roles (e.g. professor, student), personal preferences (e.g. the preferred light intensity in a presentation room) and activities (e.g., a presentation, a meeting). Our generic ontology extends the one proposed by Felicíssimo in [76] and considers six basic classes (or concepts) that represent separate contextual scopes: *Person*, *Device*, *Environment*, *Organization*, *Role* and *Activity*.

Class *Environment* describes generic physical spaces, e.g., places such as buildings or rooms. As such, subclasses of *Environment* may describe specific kinds of spaces that are common to different organizations, such as a *Classroom* or *Office*, for example. Class *Device* describes the characteristics of the computational devices. Its mandatory subclasses are *Mobile Device*, which may comprise subclasses such as *PDA*, *Smartphone*, *Netbook*, etc, and *Fixed Device*, that describes a stationary host. The class *Organization* describes some social structure or institution, like a university or a company, that may have as subclass a department, or an admission office, for example. Class *Role* describes some social or professional function attached to a given individual while class *Person* describes the personal characteristics and preferences of an individual. Finally, class *Activity* describes individual or group tasks in which a person may be engaged.

Figure 2.1 shows a diagram that represents the main classes and properties of the generic ontology. While the classes represent the types of concrete elements that may be enumerated in the system, the properties qualify these elements and are intrinsically related to the context infrastructure. For example, the property *isLocatedIn* is associated with context providers that are capable of determining the position of each device, while the property *isEngagedIn* expresses a situation in which a person may be involved.

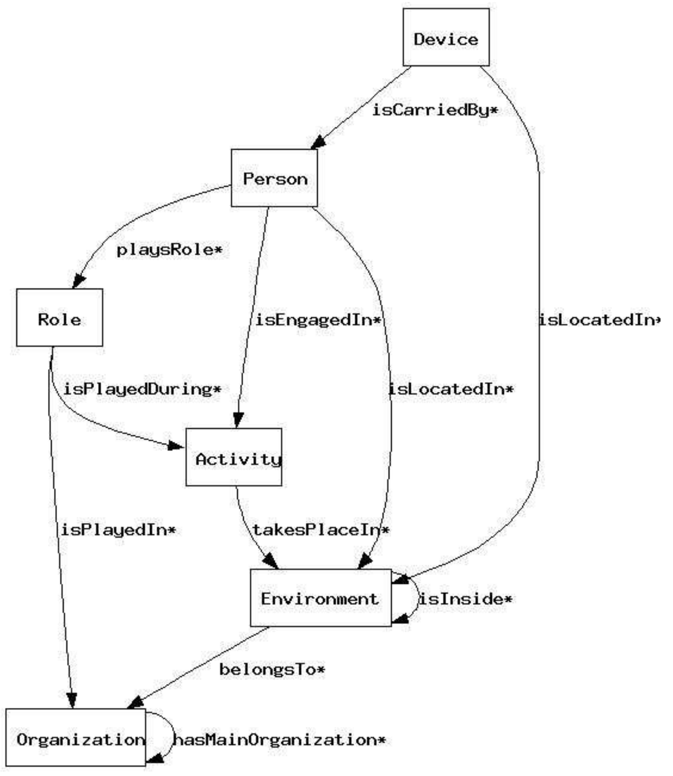


Figure 2.1: The context generic ontology

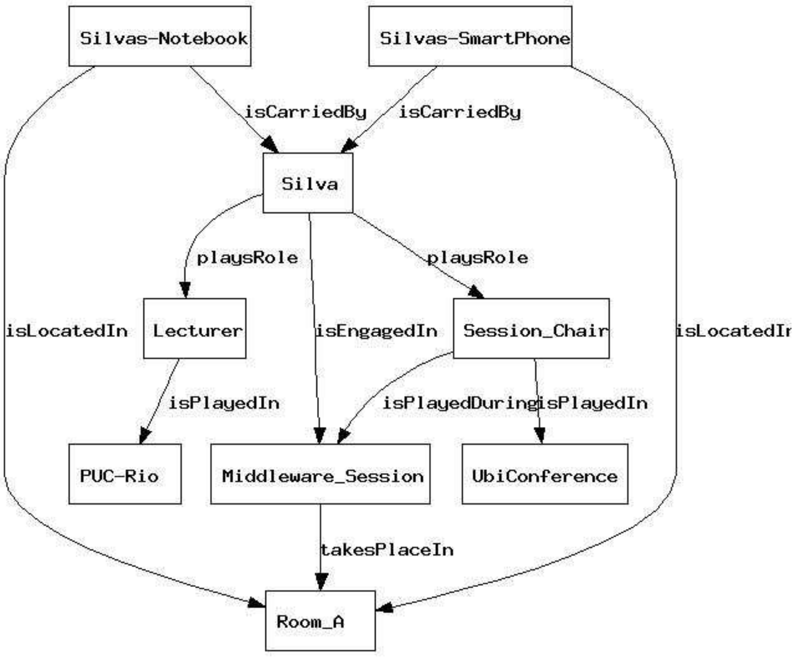
This generic ontology works as a general schema. In each case, in order to describe a particular domain or scenario, initially the generic ontology may be extended with new classes, subclasses and properties that are appropriate for the respective scenario. Then it must be instantiated with the definition of specific individuals of that domain. In our scenario, for example, first we defined the new class *Subject*, that is specific for the domain of a conference event. When describing specific elements, *Conference_Center*, *Room_A* and *Room_B* are instances of the class *Environment*. We can model *University* and *Department* as subclasses of *Organization*, and depict *PUC-Rio* as an instance of subclass *University*, and *Informatics* as an instance of subclass *Department*. The classes *Conference* and *Session* would be a subclasses of *Activity*, *UbiConference* an instance of *Conference*, and *Middleware_Session* an instance of *Session*. *Chair* would be a subclass of Role and *Middleware_Session_Chair* an instance of *Chair*, and so on.

Figure 2.2(a) shows the entities and their relations corresponding to a subset of the scenario described in Section 2.1 and according to the adopted ontology adopted in this thesis. It shows Professor *Silva* (*Person* instance), a *Lecturer* (Role instance) in the *Informatics* department of *PUC-Rio* (*Organization* instances) and also a *Programme_Committee_Member* (*Role* instance) at the *UbiConference* (*Activity* instance). In the *Middleware_Session* (*Activity* instance), which is taking place in *Room_A* (*Environment* instance), *Silva* is the *Chair* (*Role* instance). He carries with him his *Smartphone* and his *Notebook* (*Device* instances). In Fig. 2.2(b) we see the detailed description of the *Panel_Session_1*, which has *Privacy* as *Subject*, but where the target audience comprises also researchers that are interested in *Security* issues.

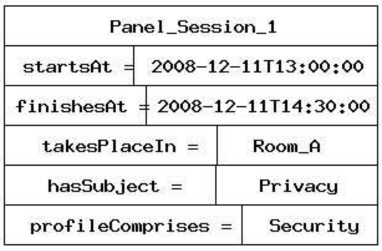
2.5 Context Reasoning

In AmI systems, as in any ubiquitous system, reasoning is required for several purposes. First, it is useful to deal with the intrinsic imperfection and uncertainty of context data. Henricksen and Indulska [24] have characterized four kinds of context imperfectness: unknown, ambiguous, imprecise and erroneous. In this case, the main tasks of reasoning are to detect possible errors, make estimates about missing values, determine the quality and validity of the context data. Second, reasoning may also be used for determining higher-level context information, i.e., to infer new, implicit context information, derived from other context data, which may be meaningful and relevant for many applications [77]. Context can be divided into lower-level (primary or raw) context and higher-level (or secondary) context. In general, lower-level context is simple and corresponds to raw data directly collected from sensors or other sources. On the other hand, higher-level context is abstract and needs to be inferred from a set of low-level context [78]. Third, reasoning is fundamental for identifying specific situations, where a situations are regarded as particular combinations of aggregated context data [29], which are relevant for triggering actions or adaptations in applications or services [30]. A situation itself can bee seen as a specific piece of higher-level context information that serves as an abstraction for application developers [26].

According to Nurmi and Floréen [79], reasoning for context-aware sys-



(a)



Panel_Session_1	
startsAt =	2008-12-11T13:00:00
finishesAt =	2008-12-11T14:30:00
takesPlaceIn =	Room_A
hasSubject =	Privacy
profileComprises =	Security

(b)

Figure 2.2: Ontology instances representing the proposed scenario, represented using the Protégé editor

tems can be approached from four main perspectives: the *low-level perspective*, which includes basic tasks such as data pre-processing, data fusion and context inference, usually performed by the sensors or the middleware, the *application-oriented perspective*, where the application can use a wide variety of reasoning methods to process the context data, the *context monitoring perspective*, where the main concern is a correct and efficient update of the knowledge base as the context changes and, finally, the *model monitoring perspective*, where the main task is to continuously evaluate and update learned context classifiers/interpreters and their models, also taking into account user feedback. Although this classification gives an interesting perspective on context reasoning, in AmI environments, we understand that, instead of four perspectives, these are in fact complementary tasks that should be present in every approach for reasoning in such context-aware systems, where context data changes dynamically and inference has to be continuously performed to trigger actions and adaptations.

The reasoning solutions are intrinsically dependent on the context model used by a system. In [33], Bikakis et al. grouped into three approaches the context reasoning solutions adopted by systems that use the ontology-based model: *ontological reasoning*, *rule-based reasoning* and *distributed reasoning. Ontological reasoning* is supported by ontology languages like OWL-DL, that can be mapped to certain classes of description logics. In this case, reasoning services are based on subsumption computation for these logics and usually include consistency and classification, as well as checking for instances of specific concepts based on their properties [80]. It may also infer knowledge from ontology axioms. In this case, a reasoner may be used to validate consistency within one ontology and "complete" the ontologies by computing implicit hierarchies and relationships based on given axioms [81].

In the *rule-based reasoning*, derivation rules are used to describe higher-level context information or specific situations based on several pieces of context information. These rules may be written in different types of logic, such as first order logic, temporal logic, description logic or fuzzy logic [23], for instance, and may be defined by the application developers or system's users, or identified using specialized techniques, such as machine learning techniques.

Distributed reasoning employs methods and techniques from the field of Distributed Artificial Intelligence to cope with the elements that collect, store, process, exchange and reason about context data in distributed, context-aware, systems [33]. These three approaches are not mutually exclusive. In fact, rule-based reasoning complements ontology-based reasoning, while both approaches are complemented by distributed reasoning techniques to deal with the physical distribution of computing and sensor devices, context providers and consumers, entities responsible for brokering and reasoning, and/or applications and users that may potentially engage into spontaneous collaboration.

Our Assumption. In this thesis, we focus only on rule-based and distributed reasoning approaches. In our scenario, the ConfOrg service, executing on the ambient infrastructure, and the ConfComp application, running on the notebook of professor Silva, rely on a two-tier context reasoning service capable of cooperatively identifying the situations (or contexts) that are of interest to the AmI application. For example, when a session is about to start and Silva is reading e-mails at the lobby, the ConfComp could open a pop up window to alert Silva that a session is going to start, and that he might want to go to the respective room. This situation may be described by an inference rule (rule-based reasoning). The overall context information that characterizes this situation is distributed among the ambient infrastructure (session schedule) and Silva's device (his preference and his location).

2.6 Rule-Based Reasoning

Derivation rules are essentially a mean for calculating a derived value of data on-the-fly, often by employing some simple kind of table lookup in a database. They are a technology often used to provide views of stored data in databases [82], i.e., describing a data query. As discussed in the previous section, rule-based reasoning consists in the use of derivation rules, based on some type of logic, to describe and infer higher-level context data or specific situations. Dey [26] presents the *situation abstraction* as a powerful feature in the design of context-aware systems: "*providing the description of the states of relevant entities in a system in the form of a* situation abstraction *requires less effort than determining which individual context components need to be contacted and determining when the collective situation has been realized or satisfied, allowing context-aware application designers to concentrate on the heart of the design process.*" As such, providing rule-based inference mechanisms may be regarded as an important requirement for middleware to support the development and deployment of AmI systems.

The use of rules to describe specific situations has also some great advantages. First, rules languages (e.g., SWRL [83], CARIN [84], RDQL [85]) provide a formal model for describing situations and performing context reasoning. As such, rule-based formalisms consist in a popular paradigm of knowledge representation [86]. The expressivity and complexity of rule languages have been studied extensively, and many decidable and tractable formalisms are known. Second, while other reasoning approaches (e.g, Bayesian or neural networks) have to be designed specifically to perform one type of inference, the use of rules brings great flexibility, as rules may be reused or easily modified by application developers to represent similar situations. Besides that, rules may be previously loaded on the start up of applications or middleware services, or even defined by means of intelligent users interfaces or learning techniques and provided dynamically. Third, rules are easy to understand and widespread used, and there are many systems that integrate them with the ontology-based model [33]. With the increasingly adoption of ontology-based context model, the use of ontological reasoning together with rule-based reasoning has gained in interest, for the addition of rules to ontological knowledge confers additional expressiveness on the context model [87]. That happens because, for reasons of decidability, DL ontological reasoning currently not allows the composition of properties. As in many applications this is a useful operation, such integration of rule-based knowledge representation and DL ontologies is currently an active area of research [62].

Our Assumption. Our DL-safe rules are based on the SWRL rule language [88]. They take the form of a conjunctive query, which consists in an implication between an antecedent (body) and a consequent (head). The intended meaning can be read as: whenever the conditions specified in the antecedent holds, then the conditions specified in the consequent must also hold. Both the antecedent and the consequent are composed of atoms, each of which represents valid unary or binary predicates in the TBox, i.e., atoms in these rules can be of the form C(x), P(x,y), where C is an class name, P is property name, and x and y are either variables or valid individual names in the ABox.

In this work, we focus on checking individuals, a fundamental reasoning task with respect to an ABox. In other words, we are concerned with the inference of class assertions and property assertions [89] to determine a set of individuals that satisfy the rule. The consequent lists variables for which the user would like to compute bindings, while the antecedent consists of atoms in which all variables from the consequent must be mentioned, but that may contain additional variables, assumed as existentially quantified. The result of the reasoning operation for such rule, i.e., the query answer, is a set of tuples representing bindings for variables mentioned in the consequent [90]. Essentially, the possible values that free variables may assume are restricted to named individuals only, confining the evaluation of such rules to the ABox. This safety condition is known as "DL-safety" and such rules are generally called "DL-safe rules." Not only are DL-safe rules decidable, but they can be solved by the available reasoner implementations [91].

Rule 2.1:

$$\textit{takesPlaceIn(?x,?y)} \wedge \textit{hasStarted(?x)} \Rightarrow \textit{isBusy(?y)}$$

Using the standard textual notation for DL rules, of the form *antecedent* $\Rightarrow$ *consequent*, an example of a rule asserting that "when session X that takes place in room Y has already started it implies that room Y is busy" would be written as in Rule 2.1, in which the variables are indicated in the standard convention of using a question mark prefix (e.g., ?x). If, for example, the ABox contained the facts *takesPlaceIn(Middleware_Session, Room_A)*, *takesPlaceIn(AmI_Session, Room_B)*, *takesPlaceIn(Privacy_Session, Room_C)*, *hasStarted(Middleware_Session)* and *hasStarted(Privacy_Session)*, the result for the rule would be the set {*Room_A*, *Room_C*}, indicating that the predicate *isBusy* is valid for both individuals.

2.7 Discussion

In this chapter, initially we described our motivating scenario, a fictitious conference that takes place in a facility enriched with AmI technology. While the attendees and organizers of the conference move through different spaces carrying one or more mobile devices, applications executing on their devices autonomously interact with different ambient services. These applications and services are context-aware and their actions are triggered based on context information collected both from the user's devices, such as his location and preferences, and the ambient infrastructure, such as the room assigned for each activity and its status. For describing this context information, we adopted an ontology-based model, in order to be able to represent not only aspects of the physical environment and computational resources but also the social aspects, such as personal preferences and activities of the users.

Applications in our scenario, such as the ConfOrg service, running on the ambient infrastructure, and the ConfComp application, running on the notebook of professor Silva, rely on a context reasoning service capable of identifying the situations in which specific actions have to be triggered. For example, when a session is about to start and Silva is reading e-mails at the lobby, the ConfComp should open a pop up window to warn Silva to go to the respective room. These situations are described using derivation rules. The context information that characterizes this situation, though, is distributed between the ambient infrastructure and Silva's device, which makes necessary the use of some distributed reasoning approach.

In the next chapter, we discuss and compare some frameworks and middleware systems that deal with distributed reasoning.