4
Cooperative Reasoning

This chapter presents our proposal of a process to perform decentralized reasoning. In the first section we explain the system model on which we base our proposal, which is formalized and explained in the following sections. Finally, we discuss design strategies to implement the service.

4.1
System Model

As we discussed in Section 2.5, in ubiquitous systems, actions or adaptations are triggered when specific situations take place, for example, a projector may be set up to show a specific set of slides when a speaker enters a conference room to give a presentation [100], or a device may be disconnected from a collaborative presentation session if the user leaves a room [101]. These situations can be described by derivation rules — represented in some type of logic — and hence may be identified by reasoning operations [31].

Most middleware systems for ubiquitous applications support rule-based inference [33], traditionally adopting a centralized approach for their reasoning mechanisms, as we argued in Chapter 3. This is the case of CoBrA [19], CHIL [22] and Semantic Space [32], for example. However, these reasoning operations may need to evaluate context data collected from distributed sources and stored in different devices, as usually not all context data is readily available to all elements of a ubiquitous environment.

In our *system model*, we consider that there are two main interacting parties in the reasoning process: the *user side* and the *ambient side*, both comprised by the services, applications and data that are available at each side. In fact, not all context information is available both at the users' mobile devices and at the ambient infrastructure. For several reasons, ranging from privacy to performance issues, some information may be available only on the *user side*, while some other information may be available only on the *ambient side*. For instance, considering our scenario (c.f. Section 2.1), at the *user side* Silva's notebook stores information about his affiliation, subjects of interest, sessions, besides his devices' resources and location. At the *ambient side* the

infrastructure stores information about the event and the environment, such as the assignment of rooms and status of each session, the status of devices (e.g. projectors that are turned on), volume of sound in the rooms, etc.

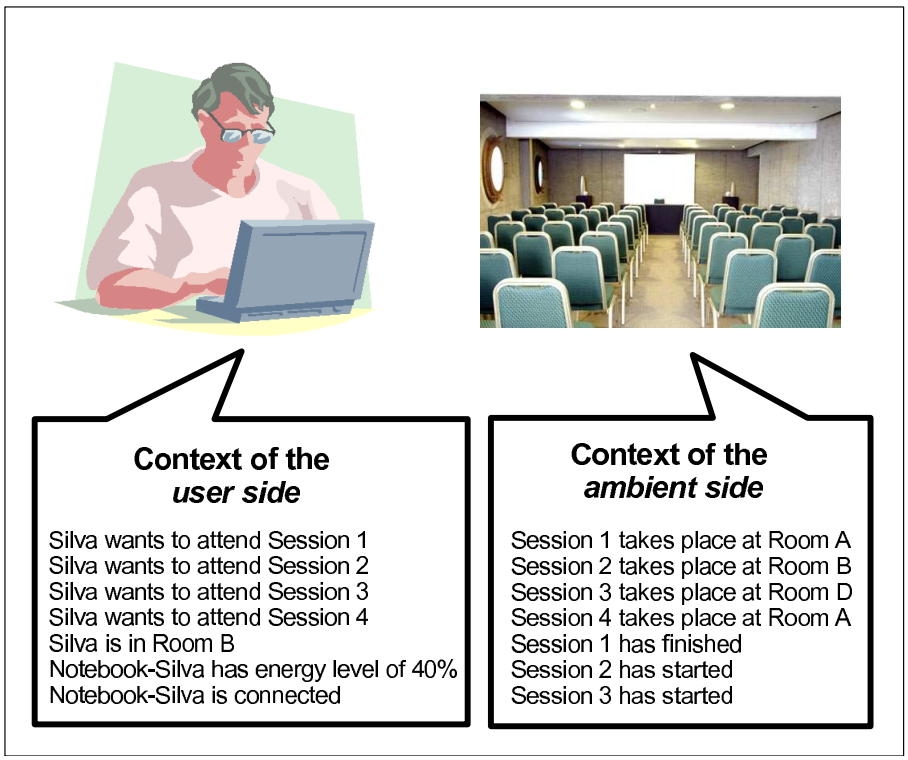

Figure 4.1: Example of context information that might be available at the user and the ambient sides of our scenario example.

Figure 4.1 shows an example of some context information that might be available on each side at a given moment. Both sides share a common context model, described Section 2.4, but have different context information, as will be formalized in Section 4.4. In the figure, we can observe that the context data related with a specific predicate, i.e., the ontology facts, are available either on the *user side*, such as "wantsToAttend", or on the *user side*, such as "takesPlace", but never on both sides. This non-overlapping condition is necessary in our two-tier model, to allow the partitioning of rules in the cooperative inference process.

We consider that applications executing on the ambient infrastructure or on the user's device rely on a rule-based reasoning service — provided by the middleware — to identify context-dependent relevant situations. A situation is described using a DL-safe rule R, in which the free variables correspond to individuals that may be available at either side. These applications may query or subscribe at the reasoning service, providing the rule R to be inferred. The result of the reasoning operation is a set of tuples S with individuals that

satisfy the rule. We assume that there is no message loss in the communication among the entities in the system.

4.2 Patterns of Interaction

Hence, for a reasoning service — running on either side — to infer relevant situations based on rules provided by applications, we devise three possible patterns of interaction: (a) *user side reasoning*, if the reasoning is performed based only on context information available at the *user side*, (b) *ambient side reasoning*, if the reasoning is performed based only on context information available at the *ambient side*, or (c) *cooperative reasoning*, when the reasoning is performed based on context information stored at both sides.

4.2.1 User Side Reasoning

Applications executing on mobile devices may be interested in situations expressed by rules in which all involved free variables represent context information available at the device. For example, in our scenario, assuming that the ConfComp was configured to warn Silva to recharge notebook when he is waiting between sessions in the lobby of the conference center. The application needs to identify the situation described by Rule 4.1, which involves device's capabilities and the user's location, the desired situation could be inferred solely based on context data available at the device. The rule states that "if Silva is located in the lobby and his mobile device's battery has less than 40% energy, then he should recharge it". In this case a reasoning service executing on the device would be able to check when the rule can be triggered.

Rule 4.1:

$$hasEnergyLevel(\text{“Notebook-Silva”}, ?c) \wedge lessThan(?c, \text{“40\%”}) \wedge isLocatedIn(\text{“Silva”}, \text{“Lobby”}) \longrightarrow shouldRecharge(\text{“Notebook-Silva”})$$

4.2.2 Ambient Side Reasoning

If all context data necessary to find the result for a rule is available to the ambient infrastructure's services, the reasoning operation can be performed entirely at the ambient side. Revisiting our scenario, let us assume that an application at the ambient side is interested in the situation described by Rule 4.2, which states that "if a conference session takes place in a given room and

the session has already started, then the room is busy". In this case a reasoning service running on the ambient infrastructure would be able to infer the rule.

Rule 4.2:

$$takesPlace(?s,?r) \wedge hasStarted(?s) \longrightarrow isBusy(?r)$$

4.2.3 Cooperative Reasoning

As a third possibility, a situation may be described by a rule containing context variables that refer to context information available both at the devices and at the ambient infrastructure. For instance, consider Rule 4.3 that states that "if Silva is located in a room where a conference session is supposed to take place and this session has already started, then Silva is busy". As shown in Table 4.1, the user's location data is available only at the user side, while the information about the sessions is available only at the ambient side.

Rule 4.3:

$$isLocatedIn(\text{“Silva”},?r) \wedge takesPlace(?s,?r) \wedge hasStarted(?s) \longrightarrow isBusy(\text{“Silva”})$$

In this case, for a reasoning service executing in the ambient infrastructure to be able to infer the rule, it would have to collect and store all context data produced in that environment — both by the ambient infrastructure itself and by all the users' devices — in a centralized way. Although there are usually no computational limitations for the ambient infrastructure related to storing or processing the large amounts of context data necessary for such reasoning, privacy issues may prevent the user from accepting the disclosure of his personal information — such as his location or personal preferences — to the ambient infrastructure.

As an option, the device at the *user side* could collect and store all context information available at the *ambient side* that would be necessary for reasoning about the proposed rules. To follow this approach, however, would entail another problem: the reasoning computation may be too heavy to be performed by the resource-limited mobile device. In fact, we can only expect that inference operations be efficiently executed on a mobile device when the corresponding rules refer exclusively to context information available at the device, rather than provided by the ambient.

We conclude that, for such types of rules, reasoning should neither be performed solely by the device, nor by the ambient infrastructure. It should rather be executed involving reasoning services on both sides, performing what we define as "cooperative reasoning". In the cooperative reasoning each side should be in charge of analyzing the context information available locally, then the outcomes of each side would have to be combined to produce an appropriate result. For this purpose, however, a strategy for cooperating in the reasoning operation needs to be defined.

4.3 Design Strategies for a Reasoning Service

In Chapter 3 we discussed the main characteristics of several reasoning approaches for distributed scenarios. We noticed that these approaches either are not completely distributed, or are not capable of inferring complex rules with variables, indicating that there must be a balance between these features. As such, in Section 4.1 we proposed our system model, which is not completely distributed, characterizing AmI systems as a two-tier model. In the previous section we discussed all possible patterns of interaction for reasoning based on this model. Considering the main functionalities of the related work and the characteristics of our model, we now discuss the functional and non-functional attributes that compose the design strategies for implementing a service to perform decentralized reasoning.

4.3.1 Functional Attributes

D1 - Support to Rule-Based Reasoning: As a first design strategy we identify the need for the service to perform inference based on rules. Rules provide not only a formal model to describe situations of interest, but also a general-purpose representation of particular combinations of context data that are relevant for ubiquitous applications or services.

D2 - Use of Variables: Free variables in rules give more flexibility to the description of situations, as the developer can refer generically to the elements of a domain, rather than mention each specifically. The result of the reasoning operation for such a rule is a set of tuples representing individuals in the ABox that bind to the variables mentioned in the consequent of the rule, like in database queries. This, however, imposes a greater complexity on the implementation of the service.

D3 - Support to Decentralized Reasoning: The reasoning service must be able to infer situations described by rules that depend on decentralized context data, i.e., context information that is distributed at the different parties of our model: the *user side* and the *ambient side*. For reasons ranging from privacy to inference performance, it may be the case that neither side has access to full context information.

D3.1 - Support to Local Reasoning: As discussed in Subsections 4.2.1 and 4.2.2, when all necessary context data is available at the reasoning side, the reasoning may be performed locally, i.e., inferred in local reasoning.

D3.2 - Support to Cooperative Reasoning: On the other hand, as discussed in Subsection 4.2.3, there should be support for *cooperative reasoning*, i.e., where services both at the *user side* and the *ambient side* are in charge of analyzing the context information available locally, and having the outcomes of each entities' local reasoning combined to determine the global inference result.

D4 - Support to Synchronous Queries: The reasoning service must be able to evaluate a rule and respond immediately to a query submitted by a client application. In case of *cooperative reasoning*, the peer reasoners have to be able to interact to produce an immediate response to the client application, as will be explained in Subsection 5.1.1. This is necessary for applications that need to check if a given situation is satisfied, e.g., "is there some event scheduled for this room I am entering?"

D5 - Support to Asynchronous Communication: This functionality is important because, due to the intrinsic limitation imposed by query-only reasoning, a client requiring up-to-date information about a situation would have to continuously poll the reasoner, causing higher load to the service and network, and draining the already scarce resources of mobile devices [102]. On the other hand, in the publish/subscribe approach the client is notified as soon as the rule is triggered, allowing a timely response of the system for the situation of interest. In the case of *cooperative reasoning*, the peer reasoners have to be able to interact to determine the result for the reasoning operation for a rule submitted by a client application each time its result changes, until this application removes the rule. This strategy will be discussed in Subsection 5.1.2.

4.3.2 Non-functional Attributes

D6 - Scalability: The decentralized reasoning service must be able to work efficiently in scenarios where a great number of users — through their mobile devices — interact with the ambient services. In such environments, many interests of client applications request the reasoning services, which have to be able to manage a great mumber of subscriptions with different rules.

D7 - Response Time: The reasoning service must be able to detect/infer situations shortly after they take place, i.e., after the context variables assume values that satisfy a rule. In particular, the latency between a context data change and its corresponding notification to the client applications must be sufficiently small for triggering the adaptations specific of that scenario.

D8 - Communication Traffic: The communication traffic involving the decentralized reasoners must not deteriorate the overall communication quality of the system.

D9 - Memory Consumption: The memory footprint of the reasoning service must be such that it can be executed on any device used in the ubiquitous system.

D10 - Robustness and Resilience: The reasoning service must be able to work correctly despite some failures of the underlying infrastructure. For that sake, the service must be able to deal with failures in the communication between the client application and the reasoner or between the different parts of the reasoner, such as message loss or content error.

D11 - Portability: As a part of the decentralized service has to be executed on the ambient infrastricture, and the other part on the users' mobile devices, the service implementation should be easily portable to different mobile or fixed devices.

In this work, we focus on two main non-functional attributes, response time (D7) and communication traffic (D8). The other attributes will be further tackled in future work. Before presenting our proposal of a reasoning process that meets the design strategies discussed above, in the following section we

formalize the cooperative reasoning operation to be able to draw an adequate strategy that solves this problem.

4.4 Cooperative Reasoning Formalization

Initially, let us assume that our domain $\mathcal{D}$ is described in a knowledge base comprised by a TBox $\mathcal{T}$, which defines all the unary and binary predicates $P \in \mathcal{P}$, i.e., classes and properties that are used to describe that domain, and an ABox $\mathcal{A}$, containing all the named individuals $I \in \mathcal{I}$ and all the asserted facts $F \in \mathcal{F}$ that represent the context data of that domain.

A rule R, decidable in $\mathcal{D}$, is composed by an antecedent R_{ant} and a consequent R_{con}, and represented as $R : R_{ant} \rightarrow R_{con}$. The antecedent R_{ant} consists of a conjunction of atoms, each in the form $P(x)$, representing an unary assertion, or in the form $P(x, y)$, representing a binary assertion, such that $P \in \mathcal{P}$, i.e., P is a predicate valid in that domain, and x and y are either individuals $I \in \mathcal{I}$ or free variables that represent such individuals.

The consequent R_{con} corresponds to an atom either in the form $P_c(x_c)$, representing an unary assertion, or in the form $P_c(x_c, y_c)$, representing a binary assertion, where P_c is a new predicate to be inferred, and x_c and y_c are either individuals defined in $\mathcal{A}$ or free variables that represent such individuals. All free variables in R_{con} must appear also in R_{ant}. This rule is represented in the Equation 4.1:

Equation 4.1:

$$R : \bigwedge P(x[, y]) \longrightarrow P_c(x_c[, y_c])$$

The result of the reasoning operation — or inference — for the rule R is a set S of tuples that represent bindings for the free variables appearing in R_{con}. The first step of the inference consists in finding a set T containing all tuples t that represent valid bindings for the free variables in R_{ant}, i.e., tuples of individuals $I \in \mathcal{I}$ that replacing each variable in R_{ant}, make it a true proposition, with each atom corresponding to a fact $F \in \mathcal{F}$. In the second step, as each free variable in R_{con} corresponds to a free variable in R_{ant}, each tuple $t \in T$ yields a tuple $s \in S$.

As an example, let us consider a TBox that defines the unary predicates P_a and P_b and the binary predicates P_c and P_d, and an ABox that contains the individuals I_1, I_2, I_3, I_4, I_5 and I_6, and the facts $P_a(I_1)$, $P_a(I_2)$, $P_a(I_3)$, $P_b(I_4)$, $P_b(I_5)$, $P_b(I_6)$, $P_c(I_1, I_3)$, $P_c(I_1, I_4)$, $P_c(I_2, I_5)$, $P_d(I_1, I_5)$, $P_d(I_2, I_6)$, and the rule R_1 as follows:

Equation 4.2:

$$R_1 : P_a(x) \bigwedge P_c(x, y) \bigwedge P_d(x, z) \longrightarrow P_{con}(y, z)$$

The result S for this reasoning operation is determined by finding the set T of all tuples t with possible values for the variables x, y and z, selected in $\{I_1, I_2, I_3, I_4, I_5, I_6\}$, such that when we replace the variables in R_{ant} by the values in a tuple t, each atom will correspond to a fact $F \in \mathcal{F}$. For example, if we pick the tuple $t = (I_1, I_4, I_5)$ and replace x, y and z in R, we have:

Equation 4.3:

$$P_a(I_1) \bigwedge P_c(I_1, I_4) \bigwedge P_d(I_1, I_5) \longrightarrow P_f(I_4, I_5).$$

In this case, each atom in R_{ant} corresponds to a fact in the ABox, as such $t \in T$ represents a valid tuple, yielding the tuple $s = (I_4, I_5) \in S$ as one of the solutions for the rule, i.e., we can infer the fact $\mathrm{P}_f(I_4, I_5) from \mathrm{R}_1$. The complete set T is $T = \{(I_1, I_3, I_5), (I_1, I_4, I_5), (I_2, I_5, I_6)\}$, and the result S for the inference of R_1 is $S = \{(I_3, I_5), (I_4, I_5), (I_5, I_6)\}$.

Let us now consider our two-tier scenario, where the context data is distributed over two different sides, one representing the *user side* and the other representing the *ambient side*. As such, our domain $\mathcal{D}'$ needs to be described as the integration of the two knowledge bases (ontologies) [73]: a local knowledge base $\mathcal{D}_L = \{\mathcal{T}_L, \mathcal{A}_L\}$, representing the local context information, i.e., on the side where we start the inference of a rule, and a remote knowledge base $\mathcal{D}_R = \{\mathcal{T}_R, \mathcal{A}_R\}$, representing the remote context information. The overall TBox $\mathcal{T}'$ consists in the integration of $\mathcal{T}_L$ and $\mathcal{T}_R$, which define the set of local predicates $\mathcal{P}_L$ and the set of remote predicates $\mathcal{P}_R$, respectively. The overall ABox $\mathcal{A}'$ consists in the integration of $\mathcal{A}_L$, containing the set of named individuals $\mathcal{I}_L$ and the set of asserted facts $\mathcal{F}_L$ that represent the local context information, and $\mathcal{A}_R$, containing the set of named individuals $\mathcal{I}_R$ and the set of asserted facts $\mathcal{F}_R$ that represent the context information on the remote side. In our model, where predicates and facts are associated with context providers and the respective provided information, the local and remote side do not share these elements, such that $\mathcal{P}_L \bigcap \mathcal{P}_R = \emptyset$ and $\mathcal{F}_L \bigcap \mathcal{F}_R = \emptyset$. On the other hand, some individuals may be present on both sides, such that $\mathcal{I}_L \bigcap \mathcal{I}_R \neq \emptyset$.

Given a rule $R' : R_{ant} \longrightarrow R_{con}$, decidable in $\mathcal{D}'$, the antecedent R_{ant} consists in the conjunction of atoms in the form $P(x)$ or $P(x, y)$, such that $P \in \mathcal{P}_L$ or $P \in \mathcal{P}_R$, and x and y may be individuals $I \in \mathcal{I}_L \bigcup \mathcal{I}_L$ or

free variables that represent these individuals. We can rewrite R_{ant} as the conjunction of two parts, a local part R_L, containing all the atoms of R_{ant} where $P \in \mathcal{P}_L$, and a remote part R_R, containing all the atoms of R_{ant} where $P \in \mathcal{P}_R$, such that R' may be represented as in the Equation 4.4:

Equation 4.4:

$$R' : R_L \bigwedge R_R \longrightarrow P_c(x_c[, y_c]), \text{ where } R_L = \bigwedge P_L(x[, y]) \text{ and } R_R = \bigwedge P_R(x[, y])$$

In the *cooperative reasoning*, we want to reason about R' without performing an integration of the local and remote knowledge bases. As such, R_L and R_R have to be evaluated separately, in different — but complementary — operations. First, in the *local reasoning* we find a set T_L of tuples t_L that represent valid bindings for the free variables in R_L, i.e., individuals $I \in \mathcal{I}_L$ that replacing the variables in R_L will make each atom correspond to a fact $F \in \mathcal{F}_L$. As R_L may have some variables in common with R_R, which we will define as the set V, this partial result bounds the next reasoning operation. In the *remote reasoning* we will have to find a set T_R of tuples t_R representing valid bindings for the variables in R_R, such that each tuple t_R has the same values for all variables $v \in V$ that appear in at least one tuple $t_L \in T_L$. Each pair t_L and t_R in which the variables $v \in V$ have the same value is a correlated pair, and the final set T of bindings for the all free variables in R_{ant} corresponds to the combination (merge) of each correlated pair. Finally, each tuple $t \in T$ yields a tuple $s \in S$.

For example, let us consider a local side where a TBox defines the unary predicate P_a and binary predicates P_c, and an ABox contains the individuals I_1, I_2, I_3, I_4, I_5, I_6 and I_7, and the facts $P_a(I_1)$, $P_a(I_3)$, $P_c(I_1, I_4)$, $P_c(I_1, I_5)$, $P_c(I_2, I_6)$ and $P_c(I_3, I_7)$, and a remote side where a TBox defines the unary predicate P_b and the binary predicate P_d, and an ABox contains the individuals I_1, I_2, I_3, I_8, I_9 , I_{10} and I_{11}, and the facts $P_b(I_8)$, $P_b(I_9)$, $P_b(I_{10})$, $P_d(I_1, I_8)$, $P_d(I_1, I_9)$, $P_d(I_1, I_{11})$, $P_d(I_2, I_8)$ and $P_d(I_3, I_{10})$, and the rule R_2 as follows:

Equation 4.5:

$$R_2 : P_a(x) \bigwedge P_b(z) \bigwedge P_c(x, y) \bigwedge P_d(x, z) \longrightarrow P_f(y, z)$$

The local part of this rule is $R_L = P_a(x) \bigwedge P_c(x, y)$. First, in the *local reasoning*, we had to find the set T_L of all tuples t_L with values for (x, y) that satisfied R_L, which corresponds to $T_L = \{(I_1, I_4), (I_1, I_5), (I_3, I_7)\}$. The remote part of the rule is $R_R = P_b(z) \bigwedge P_d(x, z)$. As x is a variable common in R_L and R_R, in the *remote reasoning* we had to find the set T_R of all tuples t_R with values for (x, z) that satisfied R_L and had $x = I_1$ or $x = I_3$,

which corresponds to $T_R = \{(I_1, I_8), (I_1, I_9), (I_3, I_{10})\}$. Combining the partial results we found $T = \{(I_1, I_4, I_8), (I_1, I_5, I_8), (I_1, I_4, I_9), (I_1, I_5, I_9), (I_3, I_7, I_{10})\}$, yielding the result $S = \{(I_4, I_8), (I_5, I_8), (I_4, I_9), (I_5, I_9), (I_7, I_{10})\}$. Below, we selected the tuple $(I_1, I_4, I_9) \in T$ to rewrite R_2, replacing the variables x, y and z, and show that the tuple $(I_4, I_9) \in S$ is a possible result, i.e., is one of the facts inferred by R_2, as all atoms of R_{ant} are facts $F \in \mathcal{F}_L \bigcup \mathcal{F}_R$.

Equation 4.6:

$$P_a(I_1) \bigwedge P_b(I_9) \bigwedge P_c(I_1, I_4) \bigwedge P_d(I_1, I_9) \longrightarrow P_f(I_4, I_9)$$

Summarizing, the *cooperative reasoning* it consists in;

1. splitting the antecedent of the rule in a *local part* and a *remote part*;
2. performing the reasoning about the *local part* of the rule in the local knowledge base to find a preliminary result T;
3. performing the reasoning about the *remote part* of the rule in the remote knowledge base, bounded by the preliminary result; and
4. combine the partial results to obtain the final result S.

In the next section we propose a strategy to perform this operation as distributed algorithm in which two reasoners cooperate exchanging messages that contain the information needed.

4.5 Discussion

In this chapter, we described the possible patterns of interaction in a system where not all context information is available for the entity in charge of performing the reasoning operation. We also discussed the functional and non-functional attributes that compose the design strategies for implementing a rule-based reasoner to be executed in such scenario. After that we formalized the *cooperative reasoning* operation. In the next chapter we propose a strategy to execute this reasoning operation and describe the distributed algorithm and communication protocol necessary to perform the complete *cooperative reasoning process*.