Modeling Decentralized Information Flow in Ambient Environments

Jurriaan van Diggelen, Robbert-Jan Beun, Rogier M. van Eijk, Peter J. Werkhoven

Institute of Information and Computing Sciences
Utrecht University, the Netherlands
{jurriaan,rj,rogier}@cs.uu.nl; peter.werkhoven@tno.nl

Abstract. This paper proposes a decentralized approach for modeling information flow in ambient environments. We study how query and notification mechanisms can be used to reduce the amount of information exchanged between agents. We will propose qualitative criteria which state whether querying a concept is appropriate given the logical structure of an agent's knowledge base. Furthermore, we will propose quantitative criteria which state which concept is most likely to be most informative, given an agent's information needs and its experience with past events.

1 Introduction

Effective and efficient information sharing is of fundamental importance for ambient intelligence. On the one hand, sufficient information should be exchanged between sensors, devices and users to maximally employ the potential use of the information present in the system. On the other hand, when vast amounts of data are available, information overload becomes a serious issue. Therefore, only the relevant information must be communicated. The problem of information sharing is complicated as the different components in the system typically represent their information at different levels of abstraction. For example, a sensor may deal with low-level information about *Temperature*, whereas an inference system used for crisis management may deal with high-level concepts such as *Fire* and *Emergency*. Another complication is the openness of the system, i.e. new devices may enter and leave the system at any time. This means that the different devices should be capable of organizing their communication networks themselves.

Many approaches that aim at guiding the information flow in ambient environments adopt a centralized approach [4, 9]. One central component is assumed which collects all information and provides access to this information to all other components. Although this approach imposes a clear organization on the information flow, it also raises a number of problems. Firstly, the system becomes brittle as the functioning of the whole system is dependent on one component. Secondly, the central distributor must be able to deal with the heterogeneities of all other components in the system. This makes it very difficult to design this component, particularly because it is not known beforehand which components will constitute the system.

Therefore, we adopt a decentralized approach by treating every component in a uniform way, i.e. as a fully autonomous agent. In this way, the problem of modeling the information flow no longer needs to be addressed as a whole (as in the centralized approach), but is split up in smaller problems which are handled by the individual agents. Every agent must have sufficient communicative skills to satisfy its information needs in an environment with heterogeneous agents that represent information at different levels of abstraction. Furthermore, the agent's communicative behavior should be minimal such that as few messages as possible are exchanged between them. This is needed to prevent information overload of the agent itself and of the agents around it.

In this paper we will provide a conceptual framework in which information needs and different levels of abstraction can be clearly represented. We also discuss two communication mechanisms, i.e. query and notification requests. We investigate which queries or notification requests can best be posed to reduce the amount of exchanged messages to a minimum.

Our approach to these two issues is as follows. An agent's knowledge base is specified as a multi-context system [6], i.e. it contains multiple contexts that are related by mappings that specify translations between them. A context consists of a set of concepts regarded relevant by an agent for performing one of its tasks. These may be a concepts like *User-Location* and *User-Identity*, which contain important contextual information for a user interface agent [10]. Also, these may be concepts like *Fire* and *Emergency* which are relevant to a crisis management agent, or *Temperature* which is relevant to a temperature sensor. The agent's information needs can be precisely represented using contexts. For example, if the information needs are defined as the context that contains *Fire*, we assume that the agent desires to know whether *Fire* is true or not, at each time instance.

Typically, two agents have some contexts in common and some contexts that differ. The agents can only communicate information that is represented in a common context. This ensures that the language used by the sending agent is understood by the receiving agent. Because the agents may view their world at a different level of abstraction, information may also be represented in a non-common context. In this case, the sending agent must translate the non-common representation to a common representation to become understood by the receiving agent. Thus, the agents must be able to translate between different contexts fluently.

The central question addressed in this paper is which concepts in one context are best to query or request for notification to resolve the information needs stated in another context. We will first approach this issue by formulating several qualitative criteria. In this way, the agent can use the logical structure of its knowledge base to decide whether querying a concept is appropriate. Because the agent bases its decision on prior knowledge, these criteria are applicable from the moment the agent joins the system. We will then approach the issue by formulating several quantitative criteria. This enables the agent to use its past experience to decide which concepts are most relevant among those concepts satisfying the qualitative criteria. Because the agent bases this decision on past experience, these criteria only become applicable after the agent has been in the system for some time.

The paper is organized as follows. In Section 2, we introduce the conceptual framework. In Section 3, we discuss the qualitative criteria for selecting the best concept to query or to be notified about. Section 4 discusses the quantitative criteria. Section 5 presents a conclusion and indicates directions for further research.

2 Framework

An agent's knowledge base is represented using description logic [1]. A description logic knowledge base consists of a TBox and an ABox. The TBox stores concepts and their definitions (like an ontology), and the ABox stores sentences constructed using these concepts. The TBox represents general knowledge about a problem domain, which is not subject to changes. The ABox represents problem-specific knowledge that is subject to occasional or even continuous change [1].

Because time plays an important role in our framework, we assume that the domain of discourse is specified as a set of time instances. This means that a concept is interpreted as a set of time instances to which the concept applies. For example, if the concept *Fire* is interpreted as $\{t3, t5\}$, it means that there was fire at time instances $t3$ and $t5$. We use a special variable NOW to denote the current time instance. This can be implemented by adopting one central time reference for all agents which instantiates the agents' NOW variables with the current time.

We adopt the description logic $\mathcal{ALC}(\mathcal{D})$ [2] as a concept language. Without going into the formal semantics, we will briefly discuss its constructs. Concepts are composed using atomic concepts and concept constructors, i.e. $\sqcap$ (conjunction), $\sqcup$ (disjunction), $\neg$ (negation). For example, the concept $\mathit{Cloudy} \sqcap \neg \mathit{Rainy}$ refers to the concept *Cloudy* and not *Rainy*. Furthermore, the language contains constructs for reasoning with numbers. For example, the construct $\geq_{50}(\mathit{Temperature})$ refers to the concept that Temperature is greater than or equal to 50 degrees.

The TBox is specified as a number of inclusion axioms of the form $c \sqsubseteq d$, meaning that the interpretation of c is a subset of the interpretation of d, i.e. all instances of c are also instances of d. For example, the TBox axiom $\mathit{Cloudy} \sqsubseteq \neg \mathit{Rainy}$ means that all time instances at which it was *Cloudy* are time instances at which it was not *Rainy*. The ABox is specified as a number of membership assertions of the form $c(t)$ meaning that t is an instance of c. For example the ABox assertion $\mathit{Rainy}(t4)$ means that it is rainy at time instance $t4$. For $c(t)$ we will sometimes simply write that c is true at time instance t. For $\neg c(t)$, we will sometimes write that c is false at time t.

Different contexts are implemented by prefixing the atomic concept names with a context identifier (similar to [3]). For example, if the concepts *Rainy* and *Cloudy* are all defined within context *C1*, a TBox axiom that relates these concepts may be $\mathit{C1{:}Cloudy} \sqsubseteq \neg \mathit{C1{:}Rainy}$. TBox axioms may also be used to

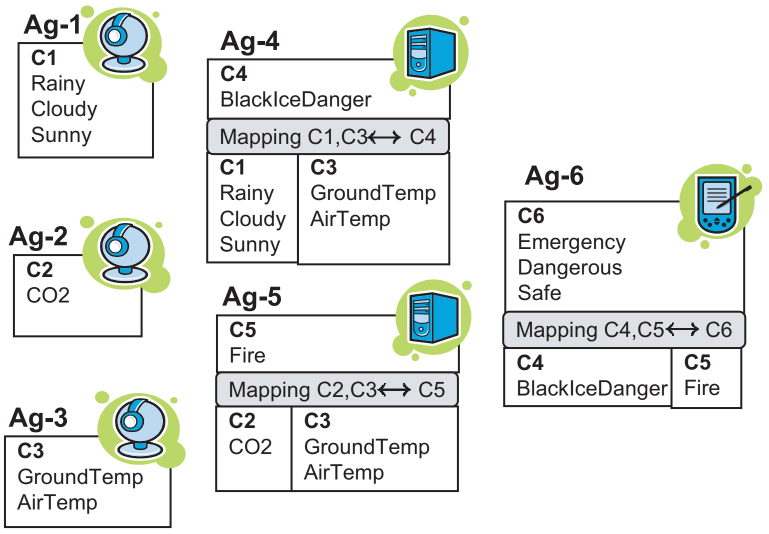

Fig. 1. Example Agents

define relations between concepts in different contexts. In this case, they are called context mappings (also known as bridge rules [6]).

Example 1.
Figure 1 illustrates six example agents. The agents' TBoxes are subdivided in different contexts which are mapped using context mappings. The TBoxes of Ag-4, Ag-5 and Ag-6 are specified below:

Ag-4
$\mathit{C1{:}Cloudy} \sqsubseteq \neg \mathit{C1{:}Rainy}$
$\mathit{C1{:}Rainy} \sqcap \leq_0(\mathit{C3{:}GroundTemp}) \sqsubseteq \mathit{C4{:}BlackIceDanger}$
$\mathit{C1{:}Sunny} \sqsubseteq \neg\ \mathit{C4{:}BlackIceDanger}$
Ag-5
$\geq_{50}(\mathit{C3{:}AirTemp}) \sqcap \geq_{10}(\mathit{C2{:}CO2}) \sqsubseteq \mathit{C5{:}Fire}$
Ag-6
$\mathit{C4{:}BlackIceDanger} \sqsubseteq \mathit{C6{:}Dangerous}$
$\mathit{C5{:}Fire} \sqsubseteq \mathit{C6{:}Emergency}$

As illustrated in this example, different agents represent their information at different levels of abstraction. They also occupy different roles in the system. The agents Ag-1, Ag-2 and Ag-3 represent low level information and perform the role of a sensor, i.e. they acquire information by sensing their environment. The agent Ag-4 is capable of processing the information produced by the sensors Ag-1 and Ag-3 (using the shared contexts C1 and C3). Likewise, Ag-5 is capable of processing the information produced by the sensors Ag-2 and Ag-3. They interpret this sensor information in terms of a higher context (C4 for Ag-4 and

C5 for Ag-5). These agents perform the role of an *aggregator* [5], i.e. they acquire information from multiple sensors and derive the consequences in terms of a higher level context. The agent Ag-5 can process the information produced by the aggregators and raises the level of abstraction in order to present it to the user, i.e. it functions as an *interpreter* [5]. One may think of Ag-6 as a PDA which, for instance, shows a green light when it is safe, a red light when it is dangerous, and a red light flashing in case of an emergency.

A description logic TBox provides proper means to model an agent's information needs [12] because it is based on an *open world assumption* [1]. This means that when a concept assertion $c(t)$ is absent in the ABox, neither $c(t)$, nor $\neg c(t)$ will be derived, i.e. the truth value remains unknown. Because the information needs apply to the current time instance NOW, we can say that the information needs on c are only fulfilled if either $c(\text{NOW})$ or $\neg c(\text{NOW})$ can be derived. In this case, we say that the agent knows-whether c. This is defined as follows.

Definition 1. *Know-whether*
An agent knows-whether c, iff $KB \models c(\text{NOW})$ *or* $KB \models \neg c(\text{NOW})$

In the above definition KB $\models$ means *it follows from the knowledge base that.* We assume that every agent has its information needs specified as a set of concepts. For example, suppose that Ag-6 has the information need *Emergency*, *Dangerous* and *Safe*. This means that it wishes to know-whether *Emergency*, *Dangerous* and *Safe*. Because the current time instance NOW increases once in a while, an agent that knows-whether a concept is true at one moment, may no longer do so after some time has passed. This causes a continuous information need for the agent.

A sensor can sense the value of a concept from its environment in order to meet its information needs. Other agents must communicate with other agents for this purpose. For example, for Ag-6 to know-whether *C6:Dangerous* is true, it may query *C4:BlackIceDanger* from Ag-4. This raises an information need for Ag-4, namely *C4:BlackIceDanger*. Subsequently, Ag-4 may query C1-concepts from Ag-1 and C3-concepts from Ag-3. In turn, this raises information needs by Ag-1 and Ag-3. Because these agents are sensors, they sense these values from the environment in order to answer the query.

Fundamental to this process is that, through the chain of queries from end user to sensor, high-level concepts are translated into lower-level concepts that can eventually be observed by sensors. To realize this reduction in information abstraction, an agent must adequately use its context mappings to translate between different contexts. The next section discusses this in further depth.

3 Qualitative criteria

Given the conceptual framework introduced in the previous section, we will regard the following question: given that an agent desires to know whether c in context C_i, which concepts d in C_j are informative?

A first class of concepts that can be readily qualified as informative contains those concepts whose membership either implies or excludes membership of concept $C_i : c$. Suppose that $C_j : d$ is such a concept. If the agent knew that $d(\textsc{now})$, the agent would know either $c(\textsc{now})$ or $\neg c(\textsc{now})$. In other words, the agent knows-whether c. Formally, this condition between $C_j : d$ and $C_i : c$ can be specified as follows: either $d \sqsubseteq c$, or $d \sqsubseteq \neg c$.

Sometimes, membership of a concept can only be decided by posing multiple queries to different agents, a process known as *query dissemination* [7]. In Example 1, *Fire* is such a concept as it must be decided using *CO2* from context C2 and *AirTemp* from context C3. Consequently, for Ag-5 to know whether *Fire*, it must query Ag-2 for *CO2* and Ag-3 for *AirTemp*. Because neither *CO2* nor *AirTemp* directly causes the agent to know-whether *Fire*, the condition discussed earlier must be generalized.

A concept d is called informative for concept c if d can be regarded as part of what must be known to exclude or conclude membership of concept c. Formally, this is defined as follows.

Definition 2. *Informative*
Concept $d \in C_j$ is informative for concept $c \in C_i$ iff there exists d' for which

- *($KB \models d \sqcap d' \sqsubseteq c$ or $KB \models d \sqcap d' \sqsubseteq \neg c$), and*
- *($KB \models d' \not\sqsubseteq c$ and $KB \models d' \not\sqsubseteq \neg c$)*

This definition states that a concept d is informative for concept c, if two conditions hold. The first condition states that, together with some other concept d' which stems from any context, d and d' must imply c or $\neg c$. The second condition states that membership of d' alone does not imply c or $\neg c$. Hence, the information about d is really necessary for the conclusion. Note that, when concept d by itself is sufficient to imply c or $\neg c$, then d also qualifies as informative. This can be easily shown be taking for concept d', the concept $\top$ (which is defined as a superconcept of all other concepts).

The idea of querying informative concepts is similar to backward chaining in expert systems [11]. To know the truth-value of a consequent, all truth-values of the conjuncts in the antecedent must be known. We would call all these conjuncts informative.

An example of the previously defined notions is given below.

Example 2. Suppose that Ag-4 in Example 1 has information need *BlackIceDanger*. We can derive the following.

- *C1:Sunny* is informative
- $\neg$*C1:Sunny* is not informative
- *C1:Rainy* is informative
- $\leq_0$(*C3:GroundTemp*) is informative
- $\geq_{50}$(*C3:AirTemp*) is not informative
- *C1:Cloudy* is not informative

3.1 Query and Notification requests

We can now state the qualitative criteria for querying a concept. These criteria state which concepts in one context are potentially useful to query for an agent that wishes to know whether a concept in another context is true.

An agent that queries a concept d does not know whether the answer will provide information that d or that $\neg d$. Therefore, if only one of the concepts d or $\neg d$ is informative, a query on concept d is appropriate. This is specified as follows.

Specification 1 *Query: Qualitative criteria*
An agent may query concept $C_j : d$ to know whether $C_i : c$ iff
- *d is informative for c or $\neg d$ is informative for c, and*
- *the agent does not know whether d.*

Note that querying a concept d to know whether c, *might* enable the agent know-whether c, but need not necessarily do so. For example, *C1:Sunny* is informative for *C4:BlackIceDanger* but $\neg$*C1:Sunny* is not. When the answer to a query on *Sunny* is "no", the agent does still not know whether *BlackIceDanger*.

Besides posing queries, a common interaction mechanism in ambient environments is a request for notification [9]. By requesting notification of a certain concept, an agent gets notified whenever that concept becomes true. The issue when it is best to query or request for notification will be addressed in Section 4.1. Here, we will be concerned with the issue which concepts are best to request for notification.

Contrary to queries, an agent that requests notification of concept d, only gets an answer when d is the case, and not when $\neg d$ is the case. When the agent did not receive any information about d, it assumes that $\neg d$ is the case. Because notification requests are intended to reduce the information exchange, the agent should anticipate on *not* receiving a message. Therefore, the negation of the concept about which it will be notified must be informative. This is formalized as follows.

Specification 2 *Request for notification: Qualitative criteria*
An agent may request for notification of $C_j : d$ to know whether $C_i : c$ iff
- *$\neg d$ is informative for c*

The criteria specified in 1 and 2 are rather loose, i.e. they do not exclude any concept that could potentially be useful to query or request for notification. Therefore, several options are left open for the agent. In Example 2, the non-informative concepts *C3:Airtemp* and *C1:Cloudy* are ruled out. The concepts *C1:Sunny*, *C1:Rainy* and *C3:Groundtemp* are all left as possible options to query. Using the logical structure of the knowledge base, it is not possible to decide which of these options is best.

For example, if the answer to a query on *Sunny* is likely to return that *Sunny* holds, this is a good query, as it immediately enables the agent to know whether *BlackIceDanger*. However, if a query on *Sunny* is likely to return that $\neg$*Sunny* holds, it may be better to query *C1:Rainy* and *C3:GroundTemp* instead. Such a decision must be based on an expectation of the answer. These quantitative issues are discussed in the following section.

4 Quantitative criteria

To take into account what a likely answer to a query will be, we will use the notion of *information gain* [8]. An agent profits most when it queries a concept with the highest information gain.

Before we will discuss information gain, we will discuss an underlying measure from information theory, i.e. *information entropy*. We will apply this measure to characterize the degree of which the truth value of a concept differs over time. A concept that is true at all time instances has entropy 0, i.e. it is maximally pure. Likewise, a concept which is false in all time instances has entropy 0. A concept that is true for half of the time instances, and false for the other half of the time instances has entropy 1, i.e. it is maximally impure. Before we give a formal definition, we introduce the following terminology:

- Δ is the domain of discourse, i.e. the set of time instances which have passed.
- $\Delta^c = \{t \in \Delta | \mathrm{KB} \models c(t)\}$ (the set of time instances at which c was true)
- $\Delta^{\neg c} = \{t \in \Delta | \mathrm{KB} \models \neg c(t)\}$ (the set of time instances at which c was false)

Information entropy can now be formalized as follows.

Definition 3. *Entropy*
$Entropy(KB, \Delta, c) = -p \log_2 p - n \log_2 n$, *where*

- $p = \frac{\#\Delta^c}{\#\Delta}$ *(the proportion of time instances at which c was true)*
- $n = \frac{\#\Delta^{\neg c}}{\#\Delta}$ *(the proportion of time instances at which c was false)*

In the above definition, # is used to denote the number of instances in a set.

Example 3. This example demonstrates how information entropy can be calculated using an agent's ABox. The table below shows the ABox of Ag-4, after eight time instances have passed. It shows which concepts are true (t) and which are false (f) at which time instances.

The entropy of Sunny can be calculated as follows. $\mathrm{Entropy}(\mathrm{KB}, \{t1..t8\}, sunny) = -\frac{1}{8}\log_2\frac{1}{8} - \frac{7}{8}\log_2\frac{7}{8} = 0.543$. The entropy of BlackIceDanger is calculated as $\mathrm{Entropy}(\mathrm{KB}, \{t1..t8\}, BlackIceDanger) = -\frac{3}{8}\log_2\frac{3}{8} - \frac{5}{8}\log_2\frac{5}{8} = 0.954$. This indicates that the concept Sunny differs less over time than the concept BlackIceDanger.

Information gain is formally defined as the expected reduction in entropy after the truth value of a concept is known.

Definition 4. *Information Gain*

$$\begin{aligned} Gain(KB, \Delta, c', c) =& Entropy(KB, \Delta, c') - \frac{\#\Delta^c}{\#\Delta} Entropy(KB, \Delta^c, c') \\ & - \frac{\#\Delta^{\neg c}}{\#\Delta} Entropy(KB, \Delta^{\neg c}, c') \end{aligned}$$

	Sunny	Rainy	$\leq_0$(GroundTemp)	BlackIceDanger
t1	f	t	f	f
t2	f	t	t	t
t3	f	f	f	f
t4	f	t	t	t
t5	t	f	f	f
t6	f	f	t	f
t7	f	t	t	t
t8	f	t	f	f

Fig. 2. The ABox of Ag-4 after 8 time instances

Example 4. Suppose that Ag-4 wishes to know whether BlackIceDanger. The information gain of Sunny can be calculated as follows. Gain(KB, $\{t1..t8\}$, *BlackIceDanger* , *Sunny*) = Entropy(KB, $\{t1..t8\}$, *BlackIceDanger*) $- \frac{1}{8}$Entropy(KB, $\{t5\}$, *BlackIceDanger*) $- \frac{7}{8}$Entropy(KB, $\{t1, t2, t3, t4, t6, t7, t8\}$, *BlackIceDanger*) = 0.954 - $\frac{1}{8}\cdot$ 0 - $\frac{7}{8}\cdot$ 0.985 = 0.092. In a similar way, it can be calculated that the information gain of Rainy is 0.348, and that the information gain of $\leq_0$(GroundTemp) is 0.584.

4.1 Query and Notification requests

To know whether a concept in context C_i is true, it is best to query a concept in C_j with the highest information gain. This idea corresponds to ID3, an algorithm for making a decision tree with as few checks as possible [8]. However, contrary to a decision tree, we only use the information gain for efficiency. The final outcome is based on the logical rules, and not on the set of training examples. For example, a query on *Sunny* is regarded as not very efficient, because it has a relatively low information gain. The best concept to query is *Groundtemp*, with the highest information gain. This idea is specified below.

Specification 3 *Query: Quantitative criteria*
Let CQ be the set of concepts that meet the qualitative criteria of querying to know whether $C_i : c$. The best concept to query is concept $d \in CQ$ for which $Gain(KB, \Delta, c, d)$ is maximal.

We will now describe the quantitative criteria for deciding whether to query or to request for notification. This decision is based on the expected answer. When the entropy among the answers is low, many queries will return the same answer. In this case, it is better to request for notification as this will reduce the information exchange. When the entropy among the answers is high, a request for notification does not substantially reduce the information flow. In this case queries are preferred. This is specified below.

Specification 4 *Query or Notify: Quantitative criteria*
Let c be a concept that matches the qualitative criteria for query and notify. If entropy of c is low and $\#\Delta^{\neg c} > \#\Delta^{c}$ then request for notification on c, else query c

For example, a request for notification on *Sunny* is preferred over a query because *Sunny* has relatively low entropy. For *GroundTemp*, a query would be preferred.

5 Conclusion and Future Research

In this paper, we have presented a decentralized approach for modeling information flow in ambient environments. In particular, we have investigated the use of queries and requests for notifications in multi-context systems. We have identified qualitative criteria, based on backtracking techniques, and quantitative criteria, based on entropy measures, for translating between different contexts. These criteria are useful to minimize the information flow between agents.

We plan to perform simulation experiments to experimentally explore the information flows that occur when every agent uses the proposed communication mechanism. Furthermore, we plan to extend this line of research by modeling more complex information needs which are based on the task models of agents. We believe that the approach presented here provides a solid basis to study the more complex interaction mechanisms that are required to deal with this scenario. Finally, we aim at studying the quality of decisions when the agents have not completely satisfied their information needs. This requires us to extend the model to take the decision making process into account.

References

1. F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Patel-Schneider, editors. *The description logic handbook: Theory, implementation and applications.* Cambridge University Press, 2003.
2. F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence, IJCAI-91*, pages 452–457, Sydney (Australia), 1991.
3. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Proceedings of the Second International Semantic Web Conference*, LNCS2870, pages 164–179. Springer Verlag, 2003.
4. H. Chen, T. Finin, and A. Joshi. An ontology for context-aware pervasive computing environments. *Knowledge Engineering Review*, 18(3):197–207, 2003.
5. A. Dey, D. Salber, and G. Abowd. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. *Human-Computer Interaction*, 16, 2001.
6. E. Giunchiglia, P. Traverso, and F. Giunchiglia. Multi-context systems as a specification framework for complex reasoning systems. *Formal Specification of Complex Reasoning Systems*, pages 45–72, 1993.
7. S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.
8. T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.
9. A. Ranganathan and R. H. Campbell. An infrastructure for context-awareness based on first order logic. *Personal Ubiquitous Computing*, 7(6):353–364, 2003.
10. B. Schilit and M. Theimer. Disseminating active map information to mobile hosts. *IEEE Network*, 8(5):22–32, 1994.
11. M. Stefik. *Introduction to knowledge systems.* Morgan Kaufmann Publishers, 1995.
12. J. van Diggelen, R. Beun, F. Dignum, R. van Eijk, and J.-J. Meyer. ANEMONE: An effective minimal ontology negotiation environment. In P. Stone and G. Weiss, editors, *Proceedings of AAMAS'06*, pages 899–906. ACM, 2006.