ICAT 2004

Towards a Semantic Contexts Maintenance Model*

Young-Tack Park, Hee-Dong Ko[+], We-Duke Cho[++]
School of Computing, Soongsil University, Seoul, South Korea
[+]IMRC, Korea Institute of Science and Technology, Seoul, South Korea
[++]Ubiquitous Frontier Office, Ministry of Science and Technology, Suwon, South Korea
park@comp.ssu.ac.kr, [+]ko@kist.re.kr, [++]chowd@ajou.ac.kr

Abstract

In ubiquitous computing, applications and services must be adapted to changing contexts in highly dynamic environments. In order to build such adaptive ubiquitous applications, maintaining the coherence of a context becomes an essential problem. A context-aware application requires an appropriate context maintenance model for maintaining the coherences of contexts. We suggest two kinds of contexts, context predicates and semantic contexts, to represent richer contexts in ubiquitous environment using ontologies. In this paper, we propose a semantic context maintenance model to provide an automatic method for maintaining coherence of semantic context databases. Semantic context maintenance model adds semantic contexts to the context model only when they are justified and automatically remove the semantic contexts if their justifications go away. Based on our semantic context maintenance model, ubiquitous applications can reason about context models free of contradictions.

Key words: Semantic Contexts, Ontology, Context Maintenance Model

1. Introduction

Context awareness is becoming an essential feature of ubiquitous services that assist our everyday lives. Dey et al [1]. defines contexts as any information that characterize a situation and that are relevant to the interaction between a user and its application. By contexts, we refer to any information from ubiquitous sensors and inferred information based on associated ontologies.

Context predicates are generated based on sensor inputs. For instance, when ubiquitous sensors detect a person entering a room, a context predicate, enter(person-id, room-number, time), is generated. These context predicates represent events occurred in ubiquitous environments. In ubiquitous computing, many RFID, UWB, IR, smart floor sensors can be used to generate context predicates when meaningful events occur.

* This research is supported in part by the Ubiquitous Autonomic Computing and Network Project, the Ministry of Science and Technology (MOST) 21st Century Frontier R & D Program in Korea

A number of context-aware systems have been developed to represent contexts explicitly [2,3]. In order to provide richer contexts, we categorize contexts into contexts predicates and semantic contexts. By contexts predicates, we mean any information that is collected from ubiquitous sensors. They may be a presence of person, status of devices, temperatures, to name a few. Contexts predicates can be essential in context model.

These context predicates are very useful for ubiquitous agents. However, we need richer semantic contexts as well. For instance, when a person enters his/her room, the context predicate, enter(person-id, room-number, time), does not represent that s/he enters his/her room. In order to represent richer contexts, we can employ domain specific inference engine but this approach seems to be limited. Instead, we propose an approach using ubiquitous ontologies.

Semantic context is proposed to represent richer contexts using relevant ontologies. Semantic contexts can provide richer high-level contextual information to ubiquitous applications. We employ context inference engines to infer ontology-based contexts based on observed context predicates. However, observed context predicates need to be removed and all the inferred semantic contexts from them need to be maintained as well. For instance, when a person enters his/her room, many semantic contexts are generated based on ontological inferences. When the person leaves the room, it is necessary to retract all the semantic contexts generated. So, we need a system to maintain the consistencies of semantic contexts. Contexts predicates are labeled “in” or “out” to indicate whether the observed context is believed in the situation. For instance, when a person enters a room, the observed context node, person_enter_the_room, becomes “in”. Later, if the person leaves the room, the observed context, person_enter_the_room, becomes “out” .

Ubiquitous systems need to maintain a model of the current context that can be shared by all the ubiquitous agents. If ubiquitous system maintains context model, it becomes very complex. In order to maintain coherences and consistent inferred contexts, we propose a semantic context maintenance model that is a collection of automatic methods for maintaining the contexts. Semantic context maintenance model apply in dynamic

situation where certain sets of observed context predicates are considered contradictory and the system must make sure that the semantic context model remains free of contradiction. Semantic context maintenance model can be used within ubiquitous systems in conjunction with ubiquitous agents to manage as a context dependency network the agent's beliefs in given situations. In this paper, we describe semantic contexts and a semantic context maintenance model to maintain the semantic contexts. In section 3, we will present the overview and motivation of our works on semantic contexts and semantic context maintenance. In section 4, the detailed semantic context maintenance model is described.

2. Related Works

In this paper, we propose concepts of semantic contexts and a semantic context maintenance model. The idea comes from using ontologies in modeling contexts. Contexts become useful only when they can be used for ubiquitous agents. Contexts play a role as a producer and ubiquitous agents consume contextual information to provide useful services to persons in ubiquitous environments. It is widely accepted using ontologies in ubiquitous agents. In previous works, context models using ontologies have been proposed and ontologies are supposed to ease shared understanding about contexts between different systems.

In Gaia [4], a context model is based on context predicates. They use first order predicate logic to represent contexts. An example context predicate in Gaia is location(chris, entering, room3231). In order to represent higher-level contexts, they employ rule-based context synthesizer. Gaia uses ontologies for checking validity of context predicates not for providing higher-level contexts. The structures of different context predicates are specified in ontology. Anand et al. [4] used the ontology to check the validity of context predicates rather than describe richer context information.

Confab's data model [6] is used to represent contextual information, such as one's location or activity. Logical storage units, called InfoSpaces, store context data about people, places, things, and services. Sources of context data, such as sensors, can populate infospaces to make their data available for use and retrieval. In Confab, context tuples are used to represent static pieces of contextual information as well as dynamic contextual information. Context tuples seem to be a way to represent contextual information without using ontologies. In addition, context tuple can also optionally have a privacy tag because Confab was designed to be effective in helping end-users manage their privacy.

Harry Chen et al [3] uses the COBRA ontology for enabling knowledge sharing and ontology reasoning based on OWL language. Harry Chen's reasoning engine is responsible for reasoning with ontology knowledge that is static knowledge and contextual knowledge. The former is derived from the underlying ontology model and the latter is dynamic knowledge that is inferred from acquired situational information. Our approach is similar to Harry Chen's but is more modular in ontological reasoning. Harry Chen's context broker agents reason about the situational contexts using ontological reasoning. Each agent has its own reasoning space and some redundant reasoning might be done in separate agent spaces that share the same reasoning steps. Instead, our approach uses common semantic contexts to be used by agents and can resolve the problem.

We use ontologies to generate semantic contexts in ubiquitous systems. Semantic contexts represent higher-level contexts as in Gaia. While Gaia uses rule-based context synthesizer, we employ ontologies that provide partially pre-defined background knowledge in generating contexts in real time. Ontology provides a vocabulary for representing objects and describing situations in ubiquitous environments. Context ontology defines a common vocabulary to share context information. Context ontology is used to describe ubiquitous dynamic events by instantiating appropriate ontology instances automatically. Semantic context generator generates semantic contexts based on ontology instances from observed context predicates. We use Web Ontology Language OWL to infer semantic contexts. Ontologies provide richer explicit semantic meanings of observed situations. To represent observed situations, we build diverse ontologies such as human, space, device, etc. Temporal ontology is used to describe temporal events and temporal relations between events.

3. Overview of Semantic Context Maintenance

Semantic context generator needs an ability to hypothetically infer semantic contexts that may be retracted later. Semantic context generator requires bookkeeping process to maintain the consistency of semantic contexts in real time. In ubiquitous environments, many events occur and fade away.

When an event occurs, a context predicate is generated and many semantic contexts are inferred. It is necessary to retract inferred semantic contexts later if the event disappears. Figure 1 shows the overview of architecture we propose for semantic context awareness and semantic context maintenance.

The fundamental architectural observation is that the semantic context awareness engine can be decomposed into two parts: a semantic context generator and a semantic context maintenance system. This partitioning allows the semantic context generator to focus on drawing ontological inferences and the semantic context maintenance model to focus on consistency of semantic

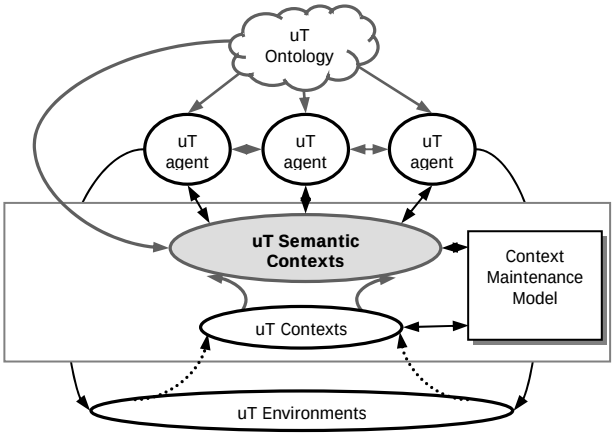

Fig. 1 Overview of Semantic Context Maintenance

contexts. Semantic context maintenance system employs an assumption-based truth maintenance system [5] used in artificial intelligence. During context generation, the semantic context generator and the semantic context maintenance system continuously interact in a well-defined protocol. Every important inference made by the semantic context generator is communicated to the semantic context maintenance system as a justification. Semantic context maintenance system records the justifications and resolves the undo problems in context generation.

Ontology axioms define subsumption relations between ubiquitous classes. Temporal and spatial axioms are used to represent temporal and spatial relations between ubiquitous events. Basically, context inference engines use backward inferences to answer the requests by agents. However, context inference engines triggers forward chaining rules to derive contexts useful for answer.

Whenever, context inference engine makes inferences, it passes associated axioms and ubiquitous situations to context maintenance system. The reasons we employ context maintenance system are two folds. First, in ubiquitous environments, most objects are mobile. For instances, human and ubiquitous objects move in and out and again they come in. When they move in, context inference engine generates all the relevant contextual information by triggering context inference axioms. And, if they move out, all the inferred contextual information must be removed from the model. However, when they move in again, the same things have to be repeated. So, we need to keep context dependency structure to enhance the performance of context inference engines. Second, context information is supposed to be shared by ubiquitous agents. Context maintenance system maintains observed and inferred contexts free of contradiction to be used by many agents. If the context maintenance system receives requests from an agent which were processed before for another agents, the system can provide answer to the agent without reasoning from the scratch.

4. Semantic Context Awareness

Semantic context represents richer high level contextual information from context predicates based on ontological reasoning. Figure 2 describes the basic idea of semantic contexts. Context predicates are generated from ubiquitous sensors. As shown in Figure 2, a context predicate, enter(Kim, Room101), is generated. Semantic context generator fetches associated ontology instances to reason about current situation. In this example, two ontology instances, Kim and Room 101, are retrieved for generating semantic contexts. We assume that Choi already entered the room.

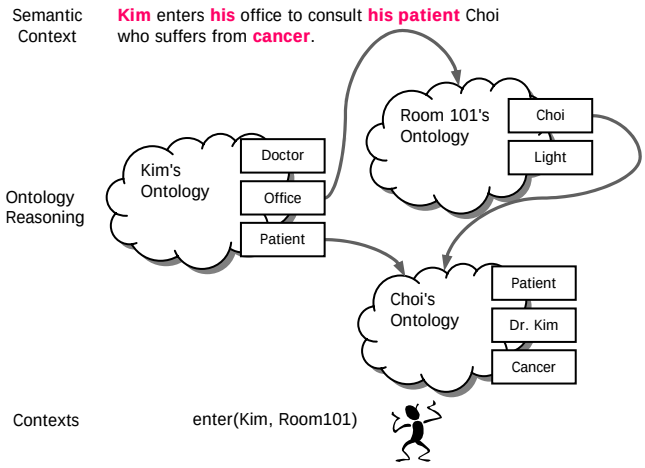

Fig. 2 Semantic Context Awareness

Based on given context predicate and retrieved associated ontologism, semantic context generator infers semantic contexts. In this example, semantic context generator learns from Kim's ontology that Mr. Kim is a doctor, his office is room 101, and his patient is Mr. Choi. Similarly, semantic context generator learns that Mr. Choi is in the room 101 and the temperature of room 101 as well. Initially, ontology instance is a partially instantiated background information. And, as ubiquitous sensors provide information to the context generator, related ontology instances become instantiated. Initially, room 101 ontology instance is partially instantiated. However, RFID sensors may detect the presence of Mr. Choi's Smart tags and semantic context generator instantiate the occupant slots of the room 101 ontology instances.

We assume that semantic contexts are a collection of high-level contexts used by ubiquitous agents to provide personal services to humans. Ubiquitous agents may use different their own reasoning mechanisms, but inform semantic context generator of required semantic context information. Semantic context generator collects all the required vocabularies for servicing ubiquitous agents and infers semantic contexts based on context predicates and ontologies in real time. Hence, our approach uses backward inferences from a set of given semantic context vocabularies. Whenever semantic context

generator generates a semantic context, it looks up index files to identify relevant ubiquitous agents and triggers the agents. This architecture enhances the performance of ubiquitous agents and alleviates the redundant inferences made by each agent otherwise.

As given a context predicate, enter(Kim, Room101), the semantic context generator fetches Kim's and Room 101's ontology instances. The semantic context generator translates ontology instances into predicate forms. This translation occurs when a new ontology instance is introduced and existing ontology instance is partially instantiated as sensors input new information. When sensors provide new information, the semantic context generator partially instantiates related ontology instances in real time. Our system uses Jena [7] to translate OWL instances into predicate forms.

In the example, the semantic context generator generates the following semantic contexts:

```
status(Kim, Doctor).   status(Choi, Patient).
office(Kim, Room101).  doctor(Choi, Kim).
patient(Kim, Choi).    suffer(Choi, Cancer)
```

And, ubiquitous agents inform semantic context generator of semantic contexts such as doctor_in_his_office and patient_in_his_doctor_office. For instance, ubiquitous service agents, called doctor agents, need to check whether doctor_in_his_office to provide comfortable environments for the doctors who enter his/her office. And, another agent may fetch patient records to be examined by doctors when patient_in_his_doctor_office context information is detected. The semantic context generator collects such semantic context information from ubiquitous agents and reason about lower level context information to infer higher level context information.

As described above, the semantic context generator fetches Kim's ontology instance. An ontology instance has static and dynamic context data slots. Static context data does not change and dynamic context data changes in real time. For instance, the status slot of Dr. Kim's ontology instance and the owner slot of Room101's ontology instance are static context data slot. These informations play background knowledge in semantic context reasoning. And, the ownerin slot and the occupant slot of Room101's ontology instance are dynamic context data. The semantic context generator fills these dynamic context data slots in real time as context predicates are generated based on inputs of ubiquitous sensors. The following is an example of OWL instance generated by semantic context generator.

```
<Doctor rdf:ID="Kim">
   <office>
     <Office rdf:ID="room101">
        <ownerin>true</ownerin>
        <occupant>
           <Patient rdf:ID="Choi">
             <suffer>Cancer</suffer>
              <status rdf:resource="#Patient"/>
              <doctor rdf:resource="#Kim"/>
           </Patient>
        </occupant>
        <occupant rdf:resource="#Kim"/>
        <owner rdf:resource="#Kim"/>
     </Office>
   </office>
   <patient rdf:resource="#Choi"/>
   <status rdf:resource="#Doctor"/>
</Doctor>
```

Semantic context generator uses backward chaining rules to infer high-level semantic contexts to be used by ubiquitous agents. In order to inference engine to reason, semantic context generator translates OWL ontology instances into predicates. Whenever a new OWL ontology instance is generated or slots of existing ontology instances are updated, the translation process is activated. Jena is used for the translation [7]. The followings are predicate forms generated by Jena (after removing OWL namespaces).

```
(type Kim Doctor)      (office Kim room101)
(type room101 Office)  (ownerin room101 true)
(occupant room101 Choi)(suffer Choi "Cancer")
(status Choi Patient)  (doctor Choi Kim)
(occupant room101 Kim) (owner room101 Kim)
(patient Kim Choi)     (status Kim Doctor)
```

These predicates are posted on the blackboard in the semantic context generator. The semantic context generator applies inference rules for generating semantic contexts that may be used by ubiquitous agents. For instance, the following Prolog rules are used to infer that a doctor enters his/her own office and a patient enters his/her doctor office.

```
doctor_in_his_office(Person, Room) :-
       type(Person, doctor),
       office(Person, Room),
       ownerin(Room, true).
patient_in_doctor_office(Person, Room) :-
       type(Person, patient),
       doctor(Doctor,Person),
       office(Doctor, Room),
       occupant(Room, Person).
```

Similarly, a situation where a doctor is consulting his/her patient can be inferred by the following rule.

```
consultation(Doctor,Patient) :-
   doctor_in_his_office(Doctor, Room),
   patient_in_doctor_office(Patient, Room).
```

Based on backward reasoning, semantic context generator generates the following semantic contexts:

doctor_in_his_office(Kim, Room101).
patient_in_doctor_office(Choi, Room101).
consultation (Kim, Choi).

Figure 3 shows dependencies between context predicates and semantic contexts in temporal dimension. Each context predicate is generated based on inputs of ubiquitous sensors. Semantic context generator generates semantic contexts based on ontologies and existing semantic contexts. Every inferences made by semantic context generator to generate semantic contexts is communicated to the context maintenance system as a justification. The justifications recorded by semantic

context maintenance system allow maintaining the integrity of semantic context model dynamically.

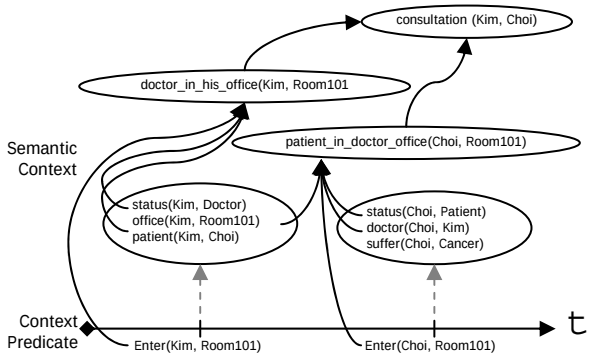

Fig. 3 Generation of Semantic Contexts

5. Semantic Context Maintenance Model

Based on context ontology and forward/backward inference rules, we generate semantic contexts. Each inference step is recorded by context maintenance system to maintain the context dependency structure. As shown in Figure 3, each semantic context element is justified by a set of justifications. For instance, semantic context element E1, consultation (Kim, Choi), is the consequent and it has two antecedents, element E2, doctor_in_his_ office (Kim, Room101), and element E3, patient_in_ doctor_office(Choi, Room101). This dependency shows that element E1 becomes true when both antecedent elements E2 and E3 are true.

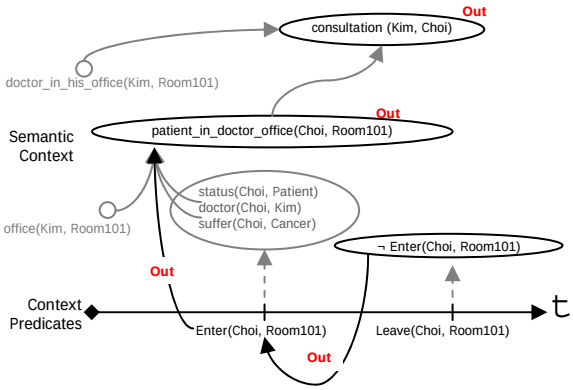

Fig. 4 Retraction of Semantic Contexts Elements

Such structure enables context maintenance system avoids “undo”. When a context predicate is popped off, it is necessary to remove all the inferred semantic contexts that were added when the observed context predicate was available. Figure 4 shows the process of undo without actual retraction of semantic contexts elements. As described above, each semantic context element is tagged by label as in ATMS [5]. When a semantic context element is no longer believed in, the label becomes out. As shown in Figure 4, suppose patient Mr. Choi leaves Room 101. The semantic context generator infers not enter(Choi, Room101) and makes the element, enter(Choi, Room101), out. By propagating the out, the semantic context generator infers that no longer consultation(Kim, Choi) is true in current contexts.

6. Representing Semantic Context Dependencies

In order to maintain dependencies between semantic context predicates, ATMS(Assumption-based Truth Maintenance System) [5,8] has been used. Figure 5 shows ATMS justification network built by the semantic context maintenance system for the above example.

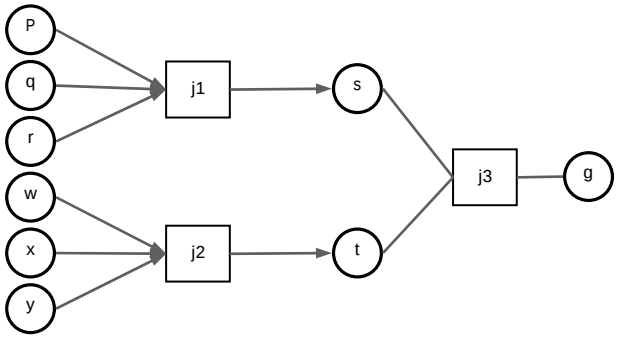

Fig. 5 ATMS Justification Network

In Figure 5, meanings of ATMS assumption nodes are as follows:

```
p:type(kim,doctor)     q:ownerin(room101,true)
r:office(kim,room101)  w:type(choi,patient)
x:doctor(kim,choi)     y:occupant(room101,choi)
s:doctor_in_his_office(kim, room101)
t:patient_in_doctor_office(choi, room101)
g:consultation(ki, choi)
```

The semantic context generator and the semantic context maintenance system continuously interact in a well-defined protocol. Every important inference made by the semantic context generator is communicated to the semantic context maintenance system as a justification. When semantic context generator fires rules in Section 4, it informs semantic context maintenance system of its rule executions with the following protocols.

```
declare_assumption(office(kim, room101)),
declare_assumption(type(kim, doctor)),
declare_assumption(ownerin(room101, true)),
atms_add_just((ownerin(room101, true),
               office(kim, room101),
               type(kim, doctor)),
       doctor_in_his_office(kim, room101)),
declare_assumption(type(choi, patient)),
declare_assumption(doctor(kim, choi)),
declare_assumption(office(kim, room101)),
declare_assumption(occupant(room101, choi)),
atms_add_just((type(choi, patient),
               doctor(kim, choi),
               office(kim, room101),
               occupant(room101,choi)),
      patient_in_doctor_office(choi, room101)),
atms_add_just((doctor_in_his_office(kim,
                          room101),
               patient_in_doctor_office(choi,
                          room101)),
      consultation(kim, choi)).
```

Semantic context maintenance model builds justification structures and beliefs as follows:

```
holds(office(kim,room101),bitvec(1)).
holds(type(kim,doctor),bitvec(2)).
holds(ownerin(room101,true),bitvec(4)).
holds(doctor_in_his_office(kim,room101),
       bitvec(7)).
holds(type(choi,patient),bitvec(8)).
holds(doctor(kim,choi),bitvec(16)).
holds(occupant(room101,choi),bitvec(32)).
holds(patient_in_doctor_office(choi,room101),
       bitvec(57)).
holds(consultation(kim,choi),bitvec(61)).
```

The notation, holds(Node,Env), means Node holds in the environment Env. Node is a node in the justification network and a context datum. Env is a set of assumptions. Bitvec(7) is another notation of bitvec(111), therefore bitvec(7) means logical OR of bitvec(1), bitvec(2), and bitvec(4). This means that the node doctor_in_his_office(kim,room101) whose Env is bitvec(7) depends on three antecedent nodes office(kim, room101), type(kim,doctor), ownerin(room101,true) whose Envs are bitvec(1), bitvec(2), and bitvec(4), respectively. As shown above, three semantic context predicates, doctor_in_his_office(kim,room101), doctor_ in_his_office(kim,room101), and consultation(kim, choi), are believed in the model.

As described in Figure 4, suppose a patient Mr. Choi leaves Room 101. This means that Mr. Choi is no more an occupant of Room 101. In order to show this dynamic situation, semantic context generator retracts the associated assumption retract_assumption(occupant (room101, choi)). Then semantic context maintenance system identifies all the related Nodes by checking bitvec database using indexed approach. The following execution segments show that Mr. Choi is no more an occupant of Room 101 and inferred facts from it are retracted as well.

```
retracting assumption ---->
    occupant(room101,choi)
retracting related node --->
    patient_in_doctor_office(choi,room101)
    consultation(kim,choi)

Semantic context maintenance model holds belief
as   follows   without   actual   deleting   the
retracted three nodes as follows.

holds(office(kim,room101),bitvec(1)).
holds(type(kim,doctor),bitvec(2)).
holds(ownerin(room101,true),bitvec(4)).
holds(doctor_in_his_office(kim,room101),
       bitvec(7)).
holds(type(choi,patient),bitvec(8)).
holds(doctor(kim,choi),bitvec(16)).
```

Later, if Mr. Choi is back to Room101, then semantic context maintenance model infers those semantic context predicates,occupant(room101,choi), patient_in_doctor _office(choi,room101), and consultation(kim,choi) without building justification from the scratch.

The context maintenance system is used in ubiquitous systems in conjunction with ubiquitous agents. Ubiquitous agents need contextual information and request information to semantic context generator in order to perform their own tasks. The context maintenance system records inferences to provide information to the semantic context generator.

8. Conclusion

We propose semantic contexts to represent high-level contexts using ontologies and the semantic context maintenance system to provide automatic methods for maintaining coherence of context databases. Semantic context generator infers high-level semantic contexts from a context predicate that represent a ubiquitous event from diverse sensors. In order to improve performance of semantic context generator, we suggest the semantic context maintenance system using an assumption-based truth maintenance system. Semantic context maintenance system records justifications of semantic contexts and adds semantic contexts to the context model only when they are justified and automatically remove inferred semantic contexts if their justifications go away. Based on our context maintenance system, ubiquitous applications can reason about semantic context models free of contradictions.

References

1. Dey,A.K., Salber, D., and Abowd, G.D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, *Human-Computer Interaction Journal*, 2001, 16(2-3)
2. Jason Hong AND James A. Landay. An Infrastructure Approach to Context-Aware Computing. *Human-Computer Interaction Journal*, 2001. 16(2-3)
3. Harry Chen AND Tim Finin AND Anupam Joshi. An Ontology for Context-Aware Pervasive Computing Environments. In *Workshop on Ontologies and Distributed Systems*, August 2003.
4. Anand Ranganathan AND Roy H. Campbell. A Middleware for Context-Aware Agents in Ubiquitous Computing Environments, *In ACM/IFIP/USENIX International Middleware Conference*, 2003, Rio de Janeiro, Brazil, June 16-20, 2003
5. Kenneth D. Forbus AND Johan De Kleer, Building Problem Solvers, MIT Press, 1993
6. Jason I. Hong AND James A. Landay, Support for Location: An Architecture for Privacy-Sensitive Ubiquitous Computing, In *Proc. of the 2nd Int. Conf. on Mobile Systems, Applications, and Services*, June 2004
7. Jena Toolkit, http://www.hpl.hp.com/semweb/
8. Yoav Shoham, Artificial Intelligence Techniques in Prolog, Morgan Kaufmann Pub., Inc, 1994