An Ontology for Context-Aware Pervasive Computing Environments*

Harry Chen, Tim Finin, and Anupam Joshi
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
{hchen4, finin, joshi}@cs.umbc.edu

Abstract

Ontologies are a key component for building open and dynamic distributed pervasive computing systems in which agents and devices share contextual information. We describe our use of the Web Ontology Language OWL and other tools for building the foundation ontology for the Context Broker Architecture (CoBrA), a new context-aware pervasive computing framework. The current version of the CoBrA ontology models the basic concepts of people, agents, places, and presentation events in an intelligent meeting room environment. It provides a vocabulary of terms for classes and properties suitable for building practical systems that model context in pervasive computing environments. We also describe our ongoing research in developing an OWL inference engine using Flora-2 and in extending the present CoBrA ontology to use the DAML spatial and temporal ontologies.

1 Introduction

Computing is moving toward pervasive, ubiquitous environments in which devices, software agents, and services are all expected to seamlessly integrate and cooperate in support of human objectives – anticipating needs, negotiating for service, acting on our behalf, and delivering services in an anywhere, any-time fashion [Weiser, 1991; Finin *et al.*, 2001]. An important next step for pervasive computing is the integration of intelligent agents that employing knowledge and reasoning to understand the local context and share this information in support of intelligent applications and interfaces. We are developing a new pervasive context-aware computing infrastructure called Context Broker Architecture (CoBrA) [Chen, 2003], to support ubiquitous agents, services and devices to behave intelligently in according to their situational contexts.

Ontologies are key requirements for building context-aware pervasive computing systems for the following reasons: (i) a common ontology enables knowledge sharing in an open and dynamic distributed systems, (ii) ontologies with well defined declarative semantics provide a means for intelligent agents to reason about contextual information, and (iii) explicitly represented ontologies allow devices and agents not expressly designed to work together to interoperate, achieving "serendipitous interoperability" [Heflin, 2003].

In the past, a number of distributed systems have been developed to support pervasive computing including the Intelligent Room [Coen, 1998], Cooltown [Kindberg and Barton, 2001], and Context Toolkit [Salber *et al.*, 1999]. These systems have made progress in various aspects of pervasive computing but are weak in supporting knowledge sharing and context reasoning. A significant source of this weakness is their lack a common ontology with explicit semantic representation [Chen *et al.*, 2001; Chen, 2003]. CoBrA provides better support for knowledge sharing and context reasoning using a common ontology defined using Semantic Web languages. In this paper, we describe the use of the Web Ontology Language OWL [van Harmelen *et al.*, 2002] and tools for building an ontology foundation in CoBrA.

In the next section, we overview CoBrA and its design rationale. In Section 5, we describe the role of the Semantic Web and the OWL language in our architecture. Section 4 describes two components that we believe to be necessary for building an ontology foundation in pervasive context-aware systems (e.g., in CoBrA). After our discussion, in Section 5, we present our initial work in building an ontology called *CoBrA Ontology* for modeling context knowledge and enabling knowledge sharing – this is the first component of the ontology foundation in CoBrA. In Section 6, we describe our on-going research work which attempts to complete the second component of the CoBrA ontology foundation, an ontology inference engine for OWL. A brief discussion of related work and our future work are given in Section 7 and Section 8, respectively. In Section 9, we summarize this document.

2 Context Broker Architecture

CoBrA is an agent based architecture for supporting context-aware computing in intelligent spaces. Intelligent spaces are physical spaces (e.g., living rooms, vehicles, corporate offices and meeting rooms) populated with intelligent systems that provide pervasive computing services to users [Kagal *et al.*, 2001]. By context, we mean an understanding of a location, its environmental attributes (e.g., noise level, light intensity,

*This work was partially supported by DARPA contract F30602-97-1-0215, NSF award 9875433, NSF award 0209001, and Hewlett Packard.

temperature and motion) and the people, devices, objects and software agents it contains.

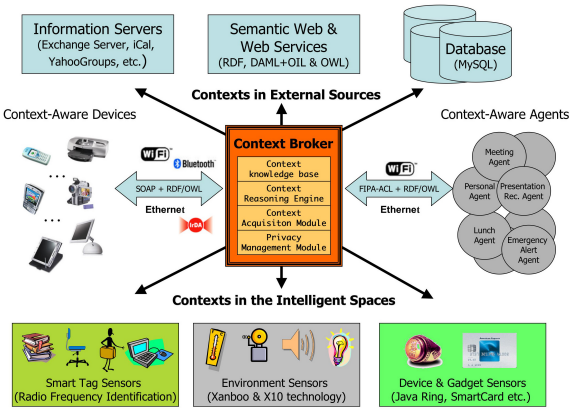

Figure 1: An intelligent context broker acquires context information from devices, agents and sensors in its environment and fuses it into a coherent model, which is then shared with the devices and their agents.

Central to our architecture is the presence of an intelligent *context broker* (or broker for short) that maintains and manages a shared model of contexts on the behalf of a community of agents (i.e., applications running on the mobile devices that a user carries or wears, services that are provided by devices in a room, and web services that provide web presences for people, places and things in the physical world [Kindberg and Barton, 2001]). In our system, a broker assumes the responsibility to (i) acquire contexts from heterogeneous information sources and maintain the consistency of the overall context knowledge through reasoning, (ii) help distributed agents to share context knowledge through the use of ontologies, agent communication languages and protocols, and (iii) protect the privacy of users by establishing and enforcing user defined policies while sharing sensitive personal information with agents in the community. Figure 1 shows a high-level design of the broker and its relationship with agents in an intelligent space.

In a large-scale intelligent space (e.g., a campus or a building), multiple brokers can form a *broker federation*. Individual broker in a federation is responsible for managing parts of the intelligent space (e.g., a room in a particular building). In a federation, brokers are related to each other in some organizational structure (e.g., peer-to-peer or hierarchical), and they can periodically exchange and synchronize context knowledge.

Our centralized broker design addresses two important issues that are key to realizing the potential of ubiquitous computing: supporting resource-limited mobile computing devices [Dertouzos, 2001; Coen, 1998; Chen and Kotz, 2000] and addressing the concerns for user privacy and security [Ackerman *et al.*, 2001; Bellotti and Sellen, 1993]. With the introduction of a context broker that operates on a stationary computer, the burdens of acquiring and reasoning over context information will be shifted away from resource-limited mobile devices to agents running on resource-rich servers; the complications inherent in establishing, monitoring and enforcing security, trust, and privacy polices will be simplified in the presence of a centralized manager. Although the existence of context broker could bring about the above advantages, its centralized design could create a "bottle neck" in a distributed system, hindering the overall system performance. In our preliminary research work, we have not addressed this problem. However, according to Kumar and Cohen [Kumar *et al.*, 2000], this "bottle neck" issue could be resolved through fault-tolerance by introducing a *persistent broker team*.

3 Rationales for Exploring Semantic Web

The responsibility of a context broker is to acquire, maintain and share a coherent and consistent model of the local context. Our approach to doing this is a knowledge-based one built on a declarative ontology of basic concepts for objects and relations in a pervasive environment. The ontology is further defined by axioms that provide additional constraints and meaning as well as rules and heuristics that can derive additional useful information. Somewhat surprisingly, we found that the languages developed for the Semantic Web are also well suited for our purpose. Key design requirements are common to the web and pervasive computing. Both are very open system with a high degree of dynamism in which independent and autonomous agents publish content and also search for information of interest.

Semantic Web is a vision of the next generation World Wide Web [Berners-Lee *et al.*, 2001]. Research efforts in the Semantic Web are driven by the need for a new knowledge representation framework to cope with the explosion of unstructured digital information on the existing Web. The present Semantic Web research focuses on the development of ontology languages and tools for constructing digital information that can be "understood" by computers [Berners-Lee *et al.*, 2001].

In the past few years, ontology language developments in the Semantic Web have converged to a new W3C standard called OWL. The OWL language shares the same root as its predecessor DAML+OIL[Connolly *et al.*, 2001] (e.g., using RDF as the modeling language to define ontological vocabularies and using XML as the surface syntax for representing information [van Harmelen *et al.*, 2002]).

We have chosen the OWL language to model context ontologies for two reasons. First, it is much more expressive than RDF or RDF-S allowing us to build more knowledge into the ontology. Second, we chose to use OWL over DAML+OIL because OWL has been designed as a standard and has the backing of a well known and regarded standards organization.

Additionally, from a system design point of view, using OWL to define context ontologies underpins two important functions of a context broker. First, it provides a means for the broker to share context knowledge with agents in an associated intelligent space. Second, it provides an ontology model which can help the broker to reason about contexts and detect knowledge inconsistency.

Knowledge sharing in pervasive context-aware systems re-

quires all agents to share a common ontology[1]. Using the OWL language, ontology concepts are defined independent from any agent implementations, and their semantics are captured using standard knowledge representation vocabularies. Taking this approach, independently developed agents can share context knowledge with the broker without pre-defined agreements on how they should interoperate.

Context reasoning is a key function of the broker. Context reasoning involves deducing context knowledge from acquired situational information and detecting inconsistency in the knowledge base. To reason about contexts, the broker can exploit ontology reasoning using logic inference engines (e.g., the DAMLJessKB [Kopena and Regli,], TRIPLE [Sintek and Decker, 2002], FaCT [Horrocks *et al.*, 1999], RACER [Volker Haarslev, 2001] and Bubo [Volz *et al.*, 2003]).

4 CoBrA Semantic Web Ontology Foundation

An ontology-driven design methodology is one way to build a distributed intelligent system (e.g., CoBrA) that can reason about contexts and can help agents to share knowledge. Using an explicit representation of the ontology, context knowledge can be reasoned over to derive additional information [Chen and Tolia, 2001], and this knowledge can also be easily share by distributed agents using standard communication languages and protocols (e.g., FIPA-ACL, KQML, and SOAP/XML-RPC). This approach requires a suitable ontological foundation on which CoBrA specific components can be built. We believe the followings are two necessary components in this foundation:

1. **Context Ontology**: The ontology provides a set of terms for describing context knowledge (i.e., explicit statements that describe contexts in the environment). The ontology should be developed in a language with appropriate expressive power and a well defined semantics. This ontology allows distributed agents to share a common understanding of the information that they exchange and to reason about additional information that is beyond what is already known.
2. **Ontology Inference Engine**: an ontology inference engine is a logic system that reasons over the semantic model of an ontology. To reason about our context ontologies in OWL, for example, an ontology inference engine should provide a set of rules for interpreting the semantic model of OWL [Patel-Schneider *et al.*, 2003] and detecting inconsistency in the knowledge base.

5 A Walkthrough of the CoBrA Ontology

This section describes key ontology concepts in the current version of the CoBrA ontology (v0.2)[2]. This ontology defines a set of vocabularies for describing people, agents, places and presentation events for supporting an intelligent meeting room system on a university campus. It also defines a set of properties and relationships that are associated with these basic concepts.

Figure 2 shows a complete list of the names of the classes and properties in the CoBrA ontology. Version v0.2 includes 41 classes (i.e., RDF resources that are type of `owl:class`) and 36 properties (i.e., RDF resources that are type of either `owl:ObjectProperty` or `owl:DatatypeProperty`).

Our ontology is categorized into four distinctive but related themes: (i) concepts that define physical places and their associated special relationships (e.g., containment relationship, social and organizational properties)[3], (ii) concepts that define agents (i.e., both human agents and software agents) and their associated attributes, (iii) concepts that describe the location contexts of an agent on a university campus, and (iv) concepts that describe the activity contexts of an agent, including the roles of speakers and audiences and their associated desires and intentions in a presentation event. In the rest of this section, we will discuss each of these four themes.

5.1 Concepts Related To Places

The notion of a place in CoBrA is restricted to a set of physical locations that are typically found on a university campus. These locations include *campus*, *building*, *room*, *hallway*, *stairway*, *restroom*, and *parking lot*. These physical locations are all assumed have well defined spatial boundaries (e.g., all locations can be uniquely identified by geographical coordinates – longitude and latitude). In addition, all locations on a university campus have identifiable string names that are assigned to them by some official bodies (e.g., by the university administration).

When modeling physical locations, we define a class called `Place` which generalizes all type of locations on a campus. This abstract class defines a set of properties that are common to all concrete physical location classes, which consists of `longitude`, `latitude` and `hasPrettyName`.

`Place` classes (including subclasses) have associated containment relationships. These relationships are defined by two related object properties[4] called `spatiallySubsumes` and `isSpatiallySubsumedBy`. The former describes the subject of this property spatially subsumes the object of this property (e.g., a building spatially subsumes a room in the building), and the latter describes the subject of this property is spatially subsumed by the object of this property (e.g., a room in the building is spatially subsumed by the building). In the context of the OWL language, these two properties are defined as an inverse property of each other.

Note that in the current version of the ontology, the domain and the range of both `spatiallySubsumes` and `isSpatiallySubsumedBy` properties are of the class type `Place`. In other word, these two properties cannot be used to make statements about the containment of a person or an agent in a physical place. However, in Section 5.2, we will describe alternative constructs for expressing this type of statements.

[1] This sharing might, in practice, be achieved with the help of ontology translation agents.

[2] A complete version of the ontology is available at `http://daml.umbc.edu/ontologies/cobra/0.2/cobra-ont` in the OWL/XML syntax.

[3] In v0.2, only containment relationship is defined, additional properties will be included in the next version of the ontology.

[4] This refers to the `owl:ObjectProperty` property

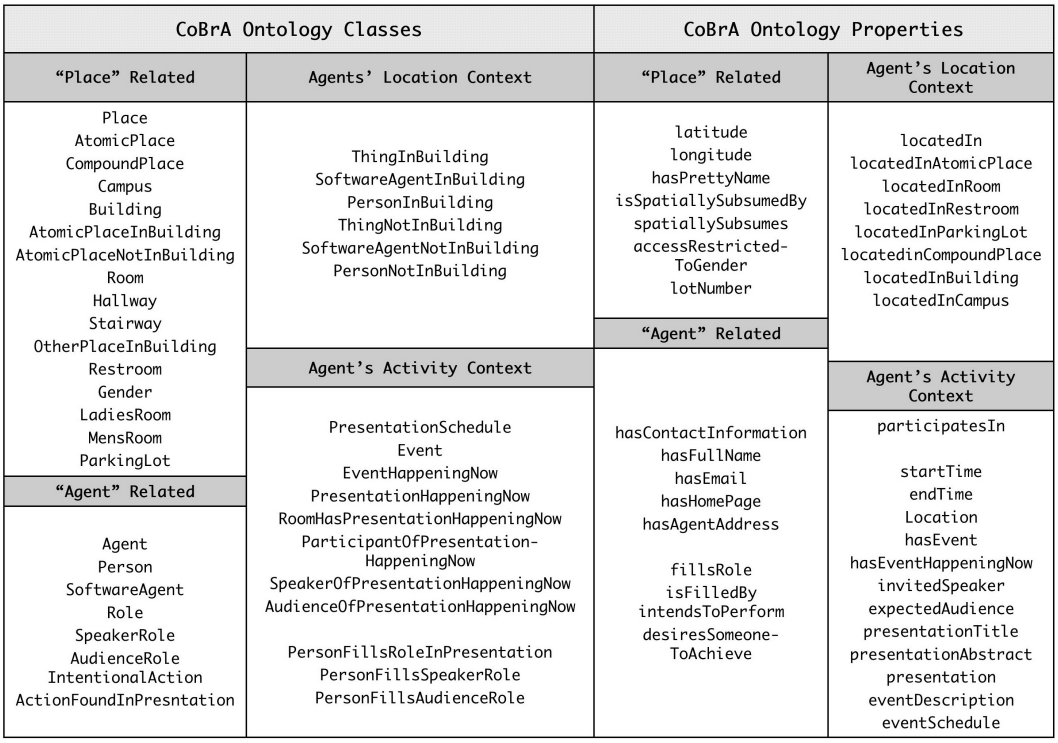

Figure 2: A complete list of the names of the classes and properties in the CoBrA ontology (v0.2).

In addition to containment relationships, physical places may be also associated with events and activities (e.g., a meeting may be taken place in a room, or an annual festive may be taken place on a university campus). To make statements about a place that is associated with some event, we introduce an object property called `hasEvent`, which has domain `Place` and range `Event`. Instances of `Event` can be associated with time intervals. We define `EventHappeningNow`, a subclass of `Event`, to represent a set of all events that are currently happening (details of this class is discussed in Section 5.4). To make statements about a place that is associated with some event that is currently happening now, we define an object property called `hasEventHappeningNow`.

AtomicPlace

Some of the concrete physical locations that we have mentioned (i.e., campus, building, room, hallway, stairway, etc.) usually do not contain (spatially subsume) other physical locations. For example, hallways, stairways and rooms in a building are not usually considered to be a type of physical place that contains other places.

For this reason, we introduce an abstract class called `AtomicPlace` to represent the set of physical places that do not contain other physical places. This class inherits all properties from its superclass `Place`. However, it restricts the range of the properties `spatiallySubsumes` and `isSpatiallySubsumedBy`. In the `AtomicPlace` class, the cardinality of the property `spatiallySubsumes` is 0, indicating all instances of this class do not contain any other physical places. The range of the property `isSpatiallySubsumedBy` is restricted to the class `CompoundPlace`, which is a subclass of `Place`. The `CompoundPlace` class represents all physical places that may contain other physical places. Figure 3 shows partial representation of these classes in OWL/XML syntax.

Some subclasses of the `AtomicPlace` class include `Room`, `Hallway`, `Stairway`, `Restroom`, `LadiesRoom`, `MensRoom` and `ParkingLot`.

CompoundPlace

While the `AtomicPlace` class is introduced to represent a set of places that contains zero number of `Place` instances, the `CompoundPlace` class is defined to represent a set of places that contains at least one or more numbers of `Place` instances. This class is also a subclass of `Place`. Being a subclass of the `Place` class, `CompoundPlace` inherits all properties from its parent class. In order to express all instances of the `CompoundPlace` class should only be spatially subsumed by instances of other `CompoundPlace`, the range of this class's property `isSpatiallySubsumedBy` is restricted to have class type `CompoundPlace`. This restriction excludes all instances of the `CompoundPlace` class to be spatially subsumed by instances of the `AtomicPlace`.

5.2 Concepts Related To Agents

The notion of an agent in CoBrA represents both humans agents and software agents. Human agents are simply users in an intelligent space. Software agents, on the other hand,

Figure 3: A partial definition of the `AtomicPlace` and `CompoundPlace` classes in OWL/XML syntax

are autonomous computing entities that provide services to users (either directly or indirectly) in an associated space.

All agents have associated properties that describe their contact information, which includes uniquely identifiable names, URLs to their home pages, and email addresses. In addition, agents are assumed to have certain roles in different events and activities (e.g., a person can have the speaker role in a presentation event, and device agents in the close vicinity may take on the presentation assistant role during the presentation session). Different roles may give rise to different desires and intentions of an agent.

In the CoBrA ontology, the notions of desire and intention are both associated with actions[5]. Specifically, the notion of desire is defined as an agent's desire for some action to be achieved by some other agents (e.g., a person with the speaker role may desire some service agents to dim the lights when his presentation starts), and the notion of intention is defined as an agent's commitment to perform some particular action (e.g., a person with the audience role may intend to download a copy of the slides after attending a presentation event).

To model ontologies for agents, we introduce a general class called `Agent`, which is a set of all human agents and computational agents. We define the class `Person` to represent human agents and the class `SoftwareAgent` to represent computational agents (both of which are subclasses of the `Agent` class and disjoints with each other). All agents in our ontology are associated with properties that describe their contact information. To generalize properties that serve as descriptions of contact information, we define an object property called `hasContactInformation`. From this property, we further define sub-properties of contact information, which consist of `hasFullName`, `hasEmail`, `hasHomePage` and `hasAgentAddress`.

Role

In our ontology, the class `Role` represents a set of all roles that are presently associated with an agent. In other words, it is an abstract class that generalizes all possible types of agent roles. In v0.2 of the ontology, pre-defined subclasses of `Role` are `SpeakerRole` and `AudienceRole`.

To associate roles with an agent, the object properties `fillsRole` and `isFilledBy` are defined. These two properties are inverse property of each other – `fillsRole` has domain `Agent` and range `Role`, and `isFilledBy` has domain `Role` and range `Agent`.

Figure 4: This is a partial definition of the concepts related to roles, intentions and desires in an intelligent meeting room system.

Intentional Actions

All actions in CoBrA are defined as instances of the class `IntentionalAction`. Informally, intentional actions are actions that an agent performs intentionally and with certain goals in mind. In our design, we assume domain applications will extend this class to define specialized subclasses and instances. To support the construction of intelligent meeting room system, we have pre-defined a set of concrete instances of `IntentionalAction` that are common in presentation events (see Figure 4).

All instances of the `IntentionalAction` class (or its subclasses) can be associated with either an instance of the `Role` class or the `Agent` class through object properties `intendsToPerform` or `desiresSomeoneToAchieve`. The domain of these two properties are union of the class `Role` and `Agent` (see Figure 4).

5.3 Concepts Related to Agent's Location Context

By location context, we mean a collection of dynamic knowledge that describes the location of an agent, which is a collection of RDF statements that describes the location property of an agent. The location property of an agent is captured through the object property `locatedIn`. It has range `Place` and domain `owl:Thing`, indicating anything (including agents) may be located in some physical place.

Physical locations, as discussed in the previous section, are categorized into two distinctive classes: `AtomicPlace` (e.g., hallways and rooms) and `CompoundPlace` (e.g., campus and building). Following the semantics of these two classes, we can make the following reasoning: *no agent can locate in two different atomic places at the same time, but an*

[5]the semantic of an action is not formal defined in the current version of the ontology. In v0.2, all instances of actions are assumed to be atomic action.

agent can be in two different compound places at the same time just in case one spatially subsumes the other. This reasoning is important for detecting inconsistent knowledge about the current location of an agent.

To capture the notion an agent can be located in an atomic and a compound place, from the `locatedIn` property we define two sub-properties called `locatedInAtomicPlace` and `locatedInCompoundPlace`. The former restricts its range to the `AtomicPlace` class, and the latter restricts its range to the `CompoundPlace` class. From these two properties, we define additional properties that further restricts the type of physical place an agent can be located in. For example, `locatedInRoom`, `locatedInRestroom` and `locatedInParkingLot` are sub-properties of `locatedInAtomicPlace`; `locatedInCampus` and `locatedInBuiding` are sub-properties of `locatedInCompoundPlace`.

For agents that are located in different places, we can categorize them in according to their location properties. For example, we define `PersonInBuilding` to represent a set of all people who are located in a building, and `SoftwareAgentInBuilding` to represent a set of software agents who are located in a building, respectively. The complement of these classes are `PersonNotInBuilding` and `SoftwareAgentNotInBuilding`.

5.4 Concepts Related to Agent's Activity Context

The activity context of an agent, similar to the location context, is a collection of dynamic knowledge about certain aspects of an agent's situational condition. While location context describes the location in which the agent is situated, activity context describes activities in which the agent participates. In our ontology, the notion of an activity is restricted to represent a set of all typical group activity events in a meeting room (meeting, presentation and discussion)[6].

Activity events are assumed have schedules. For presentation events, we define `PresentationSchedule` class to represent their schedules. Presentation schedules are defined to have `startTime`, `endTime` and `location` properties, and each of which respectively represents the start time of a presentation, the end time of a presentation and the location of a presentation event. Each presentation event has one or more invited speaker and expected audience. These two concepts are defined using the `invitedSpeaker` and `expectedAudience` properties. In addition to start time, end time and location, the schedule of a presentation usually includes a title and an abstract of the presentations. To model these, we introduce `presentationTitle` and `presentationAbstract` properties.

The activity context of an agent is usually associated with activity events that are currently happening. For example, the activity context of a speaker includes the presentation event at which he/she is giving the presentation. To model this, we introduce the `PresentationEventHappeningNow` class. This class is a subclass of the `EventHappeningNow` class which models an event with the time predicate "now".

For a given presentation that is currently happening, we can specialize the type of rooms at which the event takes place. For example, a room that has an on-going presentation event is defined as `RoomHasPresentationEventHappeningNow`, which is a subclass of `Room` and restricts the range of its `hasEventHappeningNow` property to the class `PresentationSchedule`. To describe people have speaker and audience roles in an on-going event, we define the `SpeakerOfPresentationHappeningNow` class and the `AudienceOfPresentationHappeningNow` class.

6 An OWL Inference Engine in Flora-2

In the last section, we have described the CoBrA ontology, which forms the first component in the ontology foundation in our system. In order for a context broker to reason about contexts, an inference engine for reasoning over OWL ontologies is required.

At the present, inference engines that can reason over the complete semantic model of the OWL language is still under development (in Section 5 we have mentioned a few of these emerging inference engines). As a part of our research, we are developing an OWL inference engine called F-OWL using the Flora-2 system in XSB. Flora-2 is a system that translates a unified language of F-logic, HiLog, and Transaction Logic into the XSB deductive engine [Yang and Kifer, 2002]. Flora-2 has a language syntax that is similar to TRIPLE [Sintek and Decker, 2002] and also allows ontology semantics to be defined using rules.

F-OWL is a rule-driven logic inference engine. Its implementation consists of four distinctive but related sets of rules: (i) rules that define the semantic model of the RDFS ontology language, (ii) rules that define the semantic model of the OWL ontology language, (iii) rules that draw inferences over the semantic model of RDFS, and (iv) rules that draw inferences over the semantic model of OWL. Inputs to F-OWL are collections of the N-Triple representation of some domain ontologies (e.g., context knowledge that are described using the CoBrA Ontology), and outputs from F-OWL are ontological knowledge that can be proved by the logic inferences that are defined in (iii) and (iv). To access the output ontological knowledge, Flora-2 queries can be used.

F-OWL is still in its early stage of the development. The latest version (v0.3)[7] of F-OWL support a full RDF-S inferences and partial OWL inferences (limited to the OWL-Lite sub-language constructs and some OWL Full constructs). We expect to complete a full inference of the OWL language in F-OWL by late June 2003.

7 Related Work

Our work is closely related to other pervasive and context-aware computing research such as Intelligent Room [Coen, 1998], Context Toolkit [Salber *et al.*, 1999] and Cooltown [Kindberg and Barton, 2001], One.World [Grimm *et al.*, 2000] and Centaurus [Kagal *et al.*, 2001]. In comparison to

[6] In v0.2 of the ontology, we have only included concepts related to presentation events. In the future version, we will extend the ontology to includes other activity events

[7] `http://umbc.edu/~hchen4/fowl`

the previous systems, our novel design of the context broker attempts to address challenging issues such as developing explicit ontology representations of contexts, supporting context reasoning and maintenance through logic inferences and providing user privacy protection using policies (also see discussions in Section 5).

In the previous systems, user location contexts are widely used for guiding the the decision making process of context-aware applications [Salber *et al.*, 1999; Coen, 1998; Kagal *et al.*, 2001]. However, none of them have taken advantage of the semantics of spatial relations in reasoning about contexts (i.e., information that describes the whole physical space that surrounds a particular location and its relationship to other locations).

8 Future Work

Modeling space and time are important in CoBrA. We currently have a simple model of space and spatial relationships (see Section 5.1) and an implicit representation of time and temporal relationships (see Section 5.4). In the next version of the CoBrA ontology, we plan on using, if possible, or at least mapping to, if feasible, one of the consensus ontologies for space and time.

8.1 Adopting Spatial Ontology

At present, there are two distinctive versions of spatial ontologies namely the spatial ontology in SUO [Niles and Pease, 2001] and the upper Cyc ontology [Cyc, 1997]. Recent discussions on the daml-spatial mailing list have initiated the work to develop a Semantic Web version of the spatial ontology based these ontologies[8]. The new spatial ontology will cover representations for dimension, shape, length, area, volume, latitude, longitude, elevation, political subdivisions, and topological relations (e.g., Relation Connection Calculus [Randell *et al.*, 1992]). As a short term objective, we plan to investigate the applications of Relation Connection Calculus in context reasoning (e.g., detecting inconsistency knowledge about a person locating in two places that are disconnected from each other).

8.2 Adopting Temporal Ontology

In addition to the spatial ontology, the DAML community is also developing temporal ontology for expressing temporal aspects of the contents of web resources and for expressing time-related properties of web services [Hobbs, 2002]. In this ontology, interval algebra is used to define temporal relationship axioms (after, before, inside, time-between, proper-interval, etc.) and representations for clock and calendar units (i.e., year, month, day of week, etc.).

In a short term, we plan to investigate the use of interval algebra for reasoning over the temporal relationships between different context events. For example, the relation between the `at-time(e,t)` predicate and the `during(e,T)` predicate can be used determine if a person is attending a meeting at a given time interval[9]. In an intelligent meeting room, RFID sensors periodically reports the presence of a person and describe this information using the `at-time` predicate – e.g., at 12:57 PM, they report `at-time(locatedIn(harry,room201), clock_time("12:57PM"))` and at 1:33 PM, they report `at-time(locatedIn(harry,room201), clock_time("1:33PM"))`. From this knowledge, using the interval algebra, the context broker can conclude `during(locatedIn(harry, room201), time_interval("1:00PM-1:30PM"))`.

[8] `http://www.daml.org/listarchive/daml-spatial/`

9 Conclusion

Ontologies are key requirements for building context-aware pervasive computing systems. In this paper, we have described the use of the OWL language and other tools for building an ontology foundation in CoBrA, a new pervasive context-aware architecture. With an explicit representation of context ontologies, CoBrA will allow independently developed devices and agents to interoperate and to help them to share and reason about contexts. As a part of our long term research plan, we are prototyping an intelligent context broker. Our goal is to create and deploy a pervasive context-aware meeting room in the newly constructed Information Technology and Engineering Building on the UMBC main campus[10].

References

[Ackerman *et al.*, 2001] Mark Ackerman, Trevor Darrell, and Daniel J. Weitzner. Privacy in context. *Special Issue on Context-Aware Computing. Human-Computer Interaction*, 16(2-4), 2001.

[Bellotti and Sellen, 1993] Victoria Bellotti and Abigail Sellen. Design for privacy in ubiquitous computing environments. In *Proceedings of the Third European Conference on Computer Supported Cooperative Work (ECSCW'93)*, pages 77–92. Kluwer, 1993.

[Berners-Lee *et al.*, 2001] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, may 2001.

[Chen and Kotz, 2000] Guanling Chen and David Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover, NH, nov 2000.

[Chen and Tolia, 2001] Harry Chen and Sovrin Tolia. Steps towards creating a context-aware agent system. Technical report, Hewlett Packard Labs, 2001.

[Chen *et al.*, 2001] Harry Chen, Sovrin Tolia, Craig Sayers, Tim Finin, and Anupam Joshi. Creating context-aware software agents. In *Proceedings of the First GSFC/JPL Workshop on Radical Agent Concepts*, 2001.

[9] In this example, lower case `e` represents an event instance, lower case `t` represents a time instance and upper case `T` represents a time interval.

[10] ITE building construction live feed: `http://www.cs.umbc.edu/ITE/ITE.html`

[Chen, 2003] Harry Chen. An intelligent broker architecture for context-aware systems. PhD. dissertation proposal. See http://users.ebiquity.org/~hchen4/phd/hcthsisp.pdf, 2003.

[Coen, 1998] Michael H. Coen. Design principles for intelligent environments. In *AAAI/IAAI*, pages 547–554, 1998.

[Connolly *et al.*, 2001] Dan Connolly, Frank van Harmelen, Ian Horrocks, Deb McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. *DAML+OIL (March 2001) Reference Description*, 2001.

[Cyc, 1997] Cyccorp Inc. *The Upper Cyc Ontology*, 1997. http://www.cyc.com/cyc-2-1/cover.htm.

[Dertouzos, 2001] Michael Dertouzos. *The Unfinished Revolution*. HarperCollins Publishers, 2001.

[Finin *et al.*, 2001] Tim Finin, Anupam Joshi, Lalana Kagal, Olga Ratsimore, Vlad Korolev, and Harry Chen. nformation agents for mobile and embedded devices. In *Fifth International Workshop Cooperative Information Agents*, 2001.

[Grimm *et al.*, 2000] Robert Grimm, Tom Anderson, Brian Bershad, and David Wetherall. A system architecture for pervasive computing. In *Proceedings of the 9th ACM SIGOPS European Workshop*, pages 177–182, 2000.

[Heflin, 2003] Jeff Heflin. Web ontology language (owl) use cases and requirements, 2003.

[Hobbs, 2002] Jerry R. Hobbs. A daml ontology of time. http://www.cs.rochester.edu/~ferguson/daml/daml-time-20020830.txt, 2002.

[Horrocks *et al.*, 1999] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes in Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.

[Kagal *et al.*, 2001] Lalana Kagal, Vlad Korolev, Harry Chen, Anupam Joshi, and Timothy Finin. Centaurus : A framework for intelligent services in a mobile environment. In *Proceedings of the International Workshop on Smart Appliances and Wearable Computing (IWSAWC)*, 2001.

[Kindberg and Barton, 2001] Tim Kindberg and John Barton. A Web-based nomadic computing system. *Computer Networks (Amsterdam, Netherlands: 1999)*, 35(4):443–456, 2001.

[Kopena and Regli,] Joe Kopena and William C. Regli. Damljesskb: A tool for reasoning with semantic web. edge.mcs.drexel.edu/assemblies/software/damljesskb/articles/DAMLJessKB-%2002.pdf.

[Kumar *et al.*, 2000] Sanjeev Kumar, Philip R. Cohen, and Hector J. Levesque. The adaptive agent architecture: Achieving fault-tolerance using persistent broker teams. In *Proceedings of the Fourth International Conference on Multi-Agent Systems*, pages 159–166, 2000.

[Niles and Pease, 2001] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)*, 2001.

[Patel-Schneider *et al.*, 2003] Peter F. Patel-Schneider, Patrick Hayes, and Ian Horrocks. Owl web ontology language semantics and abstract syntax. 2003.

[Randell *et al.*, 1992] David A. Randell, Zhan Cui, and Anthony G. Cohn. A spatial logic based on regions and connection. In *3rd International Conference on Knowledge Representation and Reasoning*, 1992.

[Salber *et al.*, 1999] Daniel Salber, Anind K. Dey, and Gregory D. Abowd. The context toolkit: Aiding the development of context-enabled applications. In *CHI*, pages 434–441, 1999.

[Sintek and Decker, 2002] Michael Sintek and Stefan Decker. Triple—a query, inference, and transformation language for the semantic web. In *Proceedings of International Semantic Web Conference (ISWC-02)*, 2002.

[Smith *et al.*, 2003] Michael K. Smith, Chris Welty, and Deborah McGuinness. Owl web ontology language guide. http://www.w3.org/TR/owl-guide/, 2003.

[van Harmelen *et al.*, 2002] Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider, and Lynn Andrea Stein. Owl web ontology language reference. http://www.w3.org/TR/owl-ref/), 2002.

[Volker Haarslev, 2001] Ralf Mller Volker Haarslev. Description of the racer system and its applications. In *Proceedubgs International Workshop on Description Logics (DL-2001)*, 2001.

[Volz *et al.*, 2003] Raphael Volz, Stefan Decker, and Daniel Oberle. Bubo - implementing owl in rule-based systems. 2003.

[Weiser, 1991] Marc Weiser. The computer for the 21st century. *Scientific American*, 265(30):94–104, 1991.

[Yang and Kifer, 2002] Guizhen Yang and Michael Kifer. *Flora-2: User's Manual*. Department of Computer Science, Stony Brook University, Stony Brook, 2002.