Hierarchical Situation Modeling and Reasoning for Pervasive Computing

Stephen S. Yau and Junwei Liu
Arizona State University
Tempe, AZ 85287-8809, USA
{yau, junwei.liu}@asu.edu

Abstract

Situation awareness is one of the most fundamental features of entities in pervasive computing environments to dynamically adapt their behavior to situation changes to satisfy user requirements, including security and privacy. In order to support situation-aware adaptation, it is necessary to model and specify context and situation in a way such that multiple entities can easily exchange, share and reuse their knowledge on context and situation. In this paper, an OWL-based situation ontology to model situation hierarchically to facilitate sharing and reusing of situation knowledge and logic inferences is presented. The conversion of OWL situation ontology specifications to the First-Order Logic (FOL) representations, and the performance of FOL rule-based reasoning in terms of problem size and time are discussed.

Keywords: Situation awareness, situation modeling, situation reasoning, pervasive computing, Semantic Web, ontology and OWL.

1. Introduction

In pervasive computing environments, various entities are expected to integrate and cooperate seamlessly to achieve user objectives in an anywhere-and-anytime fashion. *Situation awareness* (SAW) is the capability of the entities in pervasive computing environments to be aware of situation changes and automatically adapt themselves to such changes to satisfy user requirements, including security and privacy. SAW is one of the most fundamental features to support dynamic adaptation of entities in pervasive computing environments. A *situation* is a set of contexts in the application over a period of time that affects future system behavior. A *context* is any instantaneous, detectable, and relevant property of the environment, system, or users, such as location, available bandwidth and a user's schedule [1, 2]. A pervasive computing environment involves a set of cooperative entities, each of which has related context data. In order to support situation-aware adaptation of the entities in pervasive computing environments, it is necessary to model and specify context and situation in a way such that multiple entities can easily exchange, share and reuse their knowledge on context and situation.

With recent development of *Semantic Web* [3], ontology has been widely used to facilitate knowledge sharing and reusing. For pervasive computing environments, using ontology to model context and situation enables multiple entities to have common understanding of the context and situation during collaboration. *Web Ontology Language* (OWL) [4] is a W3C standard for representing machine interpretable knowledge in Semantic Web. OWL provides three increasingly expressive sublanguages: *OWL Lite*, *OWL DL* and *OWL Full*. OWL DL has the maximum expressiveness, and is computationally complete and decidable. In addition, it supports formal logic reasoning [4]. It is called OWL DL due to its correspondence with *Description Logics* (DL).

In this paper, an OWL-based situation ontology will be presented for context and situation knowledge modeling. The situation ontology uses OWL DL to model context and situation in a hierarchical way such that the specifications for context and situation can be easily shared and reused among multiple entities. Furthermore, the use of OWL DL supports formal logical reasoning over the situation ontology for consistency checking, subsumption reasoning, implicit knowledge inference. We will also discuss the conversion of OWL situation ontology specifications to the *First-Order Logic* (FOL) representations and the performance of FOL rule-based reasoning in terms of problem size and time.

2. Current State of the Art

Context modeling for pervasive computing has been well studied in recent years. *Composite Capabilities / Preferences Profile* (CC/PP) [5] created by the W3C, and *User Agent Profile* (UAProf) created by the WAP Forum [6] and their extensions [7, 8], are *Resource Description Framework* (RDF) based approaches that describe the context information related to the mobile devices. The *Context Broker Architecture* (CoBrA) system [9] has a set of context entities modeled using OWL, and uses broker-centric agent architecture to acquire, share, reason and protect context information. The *Aspect-Scale-Context* (ASC) model and *Context Ontology Language* (CoOL) are introduced for context modeling and can be used in DAML-S model to represent the context information of services in [10]. The *Context Ontology* (CONON) context modeling approach [11] models an upper ontology for context and uses OWL DL language to represent and reason over CONON. These approaches all focus on modeling context, which is not sufficient for situation modeling.

Among the approaches for modeling situation, the following are four major ones: In *Context Toolkit* [12], situation is modeled on a system level as the aggregation of context, but there is no language level situation modeling. *Situation calculus* and its extensions [13, 14] model situation based on the effects of actions and events, and consider situation as a complete state of the world. Hence, situation is not fully described. This causes the well-known frame problem and ramification problem [13]. *A core SAW ontology* [15] models situation as a collection of Goals, SituationObjects and Relations using UML, and can be converted to OWL representation. How to model context, derive situation based on context and reuse simple situation definition to compose complicated situation are not considered. A conceptual model for context and situation for service-based systems and a situation specification example based on the conceptual model using F-logic are presented in [16, 17].

3 Requirements of Situation Modeling

Pervasive computing environments usually involve entities with different platforms connected through heterogeneous wired and wireless networks. These entities need to interoperate with each other to achieve user objectives. To make these entities situation aware in such dynamic environments, the situation modeling approach must satisfy the following requirements:

- **Machine interpretable**: The modeled situation knowledge must be easily exchanged among various devices across heterogeneous networks. Hence, it must be machine-interpretable.
- **Semantic-based**: The modeled situation knowledge must have well-defined semantics so that multiple entities in different environments can understand and interoperate with each other correctly.
- **Reusable**: The modeled situation knowledge should be reusable to reduce the information needed to be transferred and processed.
- **Extensible**: Different pervasive computing systems will have different domain specific knowledge. The modeling approach must support extension of such domain specific knowledge.
- **Logic inference**: The modeled situation knowledge should support formal logic inference for verification and reasoning.

4 OWL-based Situation Ontology

Due to the complexity of pervasive computing environments, it is impossible to enumerate all possible contexts and situations in a single ontology. Our situation ontology only models the upper ontology for context and situation by defining a set of core classes and relations using OWL DL. For each pervasive computing application, specific domain-related knowledge can be easily extended. Consider a smart conference environment scenario: "If the user is in the conference room and the light is on, then the user is ready for a meeting". Complicated situation (*ReadyForMeeting*) is derived from the basic contexts (*location* and *light*). Based on the conceptual model presented in [16, 17], the contexts in our OWL-based situation ontology are aggregated as situations and complicate situations are composed of simple situations in a hierarchical structure to facilitate the sharing and reusing of context and situation knowledge since the semantics of context/situation specification can be clearly understood by all entities in the system. The hierarchical structure of situation ontology can be roughly divided to two layers: context layer and situation layer. By separating context layer and situation layer, we separate the context acquisition and processing from the situation evaluation, which gives a clearer view of SAW and facilitates SAW development [12].

Context layer: Context layer models context definition, contextual data of entities and various context value domains in an upper ontology with the following ontology classes and relations:

- *Context* class: The *context* class is the super class for all the contexts in pervasive computing environments. Any instance of the *context* class

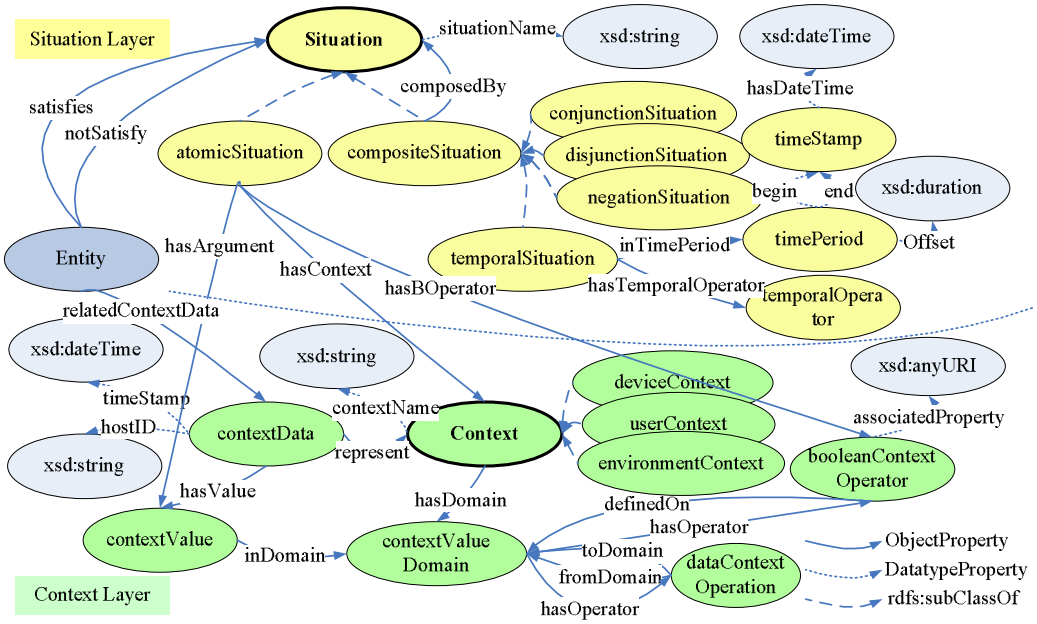

Figure 1. The situation ontology

represents a conceptual context, such as *availableBandwidthContext* or *locationContext*. Different contexts can be indexed hierarchically based on class hierarchy, such as *deviceContext*, *environmentContext* and *userContext* in Figure 1.

- *contextData* class: An instance of *contextData* will represent a certain context, and have associated host ID, timestamp and context value. All the *contextData* instances, which represent the same context and have the same host ID, define a specific context instance in the system within a certain time period. Any *Entity* in pervasive computing system can specify related contextual data using property *relatedContextData*.
- *contextValue* class: The actual value of the context is modeled by *contextValue* class. The context value in pervasive computing environments can be simple, such as the float number temperature value, or complicated data structures, like a location involving address number, street name, city, state and zip code. To support such variation, the *contextValue* class can have multiple subclasses to deal with different types of values. User can define a complex ontology class *streetLocation* class as a subclass of *contextValue* class. Simple data types like float or Boolean can be wrapped into *floatValue* or *booleanValue* classes.
- *contextValueDomain* class: For each context, the valid domain of its value is defined by *contextValueDomain* class with *hasDomain* property. Each contextValue instance also belongs to a certain context value domain via *inDomain* property. Multiple context value domains of a context corresponding to different context value scales. For example, the *available-MemoryContext* may have two context value domains corresponding to scales of MB and Bytes.
- *dataContextOperation* / *booleanContextOperator* class: The context value interpretation between different domains is modeled as *dataContext-Operation*. The *booleanContextOperor* is a special type of context value interpretation which returns a Boolean value and is used to compose atomic situation definition in the situation layer. Each *booleanContextOperor* instance may associate with a *rdf:Property* by the URI of the property, such as *owl:sameAs* or *location:sameFloor*,

Situation layer: The situation layer is built on top of the context layer to aggregate context into situations and to compose complicated situations of simple situations via logical composition and temporal composition. Whether an entity in a pervasive computing environment satisfies a certain situation or not is specified via *satisfies* and *notSatisfy* properties.

- *Situation* class: The *Situation* class is the super class for all the situations in pervasive computing environments. Different situations form a hierarchy based on their derivation and can be divided to two major categories: *atomicSituation* and *compositeSituation*.
- *atomicSituation* class: The *atomicSituation* class represents all the basic situations whose value is directly derived from the value of a single contextual data with respect to certain operator and argument. The situation layer links to the context layer with three properties: atomicSituation:

hasContext, *hasBOperator* and *hasArgument*. The evaluation of the *atomicSituation* has the semantic of "*The atomic situation is satisfied by an entity iff the entity has a related contextual data representing the context and the Boolean context operator over the contextual data value and the argument returns true.*"

- *compositeSituation* class: The *compositeSituation* class represents complicated situations: either the logical composition over other situations (*conjunctionSituation*, *disjunctionSituation* and *negationSituation*), or the *temporalSituation* whose value is derived from the value history of another situation. All these four subclasses of compositeSituation are disjoint with each other. Although supporting all the three types of logical composition is redundant, it is more convenient for situation specification. A *temporalSituation* instance must have associated temporalOperator, such as *alwaysTrue* or *onceTrue*, and timePeriod.

Using the situation ontology, the above-mentioned smart conference scenario can be specified as "situation *ReadyForMeeting* is the conjunction of two atomic situations *InConferenceRoom* and *LightOn", where InConferenceRoom* situation is *"the location-Context value is same as crLocation"* and *LightOn* situation is "*the lightContext value is true*".

5. OWL Ontology Reasoning

Various OWL ontology inferences over our situation ontology can be performed by DL reasoners, such as *RACER* [18] and *Pellet* [19]. The following inferences are commonly used:

- **Consistency checking**: Check whether a situation specification is consistent by reasoning if there is any inconsistent ontology class or inconsistent ontology instance in the specification.
- **Subsumption reasoning**: Check whether an ontology class subsumes another. This is useful to identify implicit subsume relations in situation specifications.
- **Implicit knowledge reasoning**: Reason about the implicit knowledge conveyed by situation specifications. For example, we can reason implicit knowledge "*room401 sameFloor room415*" from explicitly specified knowledge "*room401 sameFloor room402*" and "*room402 sameFloor room415*" if the *sameFloor* is a transitive property. For pervasive computing applications, it is impossible for designers and users to explicitly specify all the knowledge. With implicit knowledge reasoning capability supported by OWL, the facts needed to be specified can be greatly reduced without losing any knowledge.

6. FOL Rule-based Reasoning

Our OWL-based situation ontology also supports reasoning about whether an entity satisfies a certain situation or not by FOL rule-based reasoning. To achieve this, the situation ontology specifications are first converted to FOL representations and then FOL rule-based reasoning is performed using FOL provers.

6.1 Converting OWL Situation Specifications to FOL Representations

The basic idea of converting OWL specifications to FOL representations is to translate class references to unary predicates, properties to binary predicates, and OWL axioms (such as *owl:sameAs*) to FOL rules [20]. In [11], FOL reasoning over OWL ontology was also discussed.

To evaluation if there is any entity u satisfying a situation s, we can decompose the situation based on the situation hierarchy and generate FOL rules from the OWL situation specifications according to Table 1. By recursively performing these transformation rules, the composition tree of the situation and a set of FOL rules can be generated. For example, the composition tree and rules of *ReadyForMeeting* situation defined in Section 4 are shown in Figure 2.

Table 1. Transformation rules from OWL situation specifications to FOL rules

atomicSituation: hasContext: c hasBOperator: o hasArgument: a	p = the associated property of o $(?u\ hasContextData\ ?cd)$ $\wedge(?cd\ present\ c)$ $\wedge(?cd\ hasValue\ ?v)\wedge(?v\ p\ a)$ $\Rightarrow(?u\ satisfies\ s)$
conjunctionSituation: composedBy: $s_1, s_2, \ldots s_n$	$(?u\ satisfies\ s_1)\wedge(?u\ satisfies\ s_2)$ $\wedge\ldots\wedge(?u\ satisfies\ s_n)$ $\Rightarrow(?u\ satisfies\ s)$
disjunctionSituation: composedBy: $s_1, s_2, \ldots s_n$	$(?u\ satisfies\ s_1)\Rightarrow(?u\ satisfies\ s)$ $(?u\ satisfies\ s_2)\Rightarrow(?u\ satisfies\ s)$ ... $(?u\ satisfies\ s_n)\Rightarrow(?u\ satisfies\ s)$
negationSituation: composedBy: $\overline{s}$	$(?u\ notSatisfy\ \overline{s})\Rightarrow(?u\ satisfies\ s)$

With user-defined properties and the extension of SWRL [21], more complicated operators can be supported. For example: *OnSameFloorAsConference-*

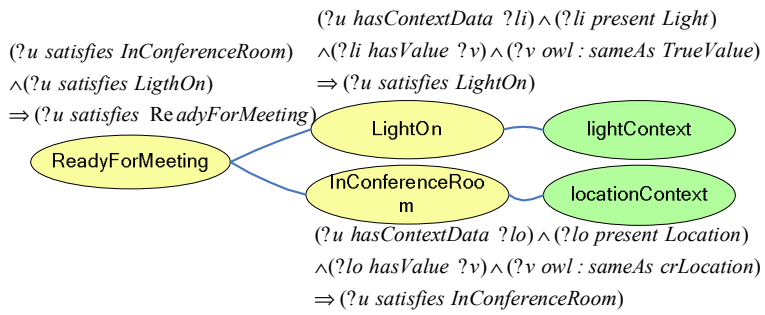

Figure 2. A composition tree example

Room situation can be defined based on the *sameFloor* property of location class, and the rule is

(?u hasContextData ?lo) ∧ (?lo present Location)
∧(?lo hasValue ?v) ∧ (?v sameFloor crLocation)
⇒ (?u satisfies OnSameFloorAsConferenceRoom)

Furthermore, SWRL build-ins can also be used to reason whether a *booleanContextOperator* returns true value. For example, SWRL building-in *swrlb:greatThanOrEqualTo* can be used to reason the situation *HasBroadBandConnection*.

(?u hasContextData ?nb)
∧(?nb present NetworkBandwidth) ∧ (?nb hasValue ?v)
∧(?v greatThanOrEqualTo "256KB")
⇒ (?u satisfies HasBroadBandConnection)
(?v inDomain $?d_1$) ∧ ("256KB" inDomain $?d_2$)
∧($?d_1$ sameAs $?d_2$) ∧ (?v hasFloatValue $?f_1$)
∧("256KB" hasFloatValue $?f_2$)
∧($?f_1$ swrlb : greatThanOrEqualTo f_2)
⇒ (?v greatThanOrEqualTo "256KB")

6.2. Experimental Results of FOL Rule-based Reasoning

In this section, the performance of FOL rule-based reasoning in terms of problem size and time will be discussed.

In our experiments, the Jena2 toolkit [22] was used to generate and parse OWL specifications and convert OWL specifications to DFG problems supported by *SPASS* FOL theorem prover [23]. The RDF triple size of the situation specifications used in our experiments varied from 56 to 1980. We compared the size of FOL problems in terms of the number of functions, predicates and formula. We also compared the average time needed for the conversion from OWL specifications to FOL problems and the average time for the execution of FOL reasoning. The hardware used in our experiments is Dell OPTIPLEX GX820 Desktop with 3GHZ CPU and 1GB memory. The results are shown in Figure 3.

From these experimental results, we have the following observations:

- The size of the FOL problem in terms of the number of functions and formula increases when the size of the OWL specification increases. It is noted that in our experiments, all the situation specifications are defined using the same ontology, and hence the number of predicates does not change.
- The time needed to convert an OWL specification to FOL problem is negligible. Even for a large OWL specification including 2000 RDF triples, the conversion only takes about 160 microseconds.
- The time needed for FOL rule-based reasoning increases dramatically when the size of the OWL specification increases, especially when the OWL specification size is over 1000 RDF triples.

These observations indicate that FOL rule-based reasoning is feasible for situation reasoning for non-time-critical applications, especially when the size of the situation specification is small. For large situation specifications, each single situation reasoning task usually only requires a small part of the specification

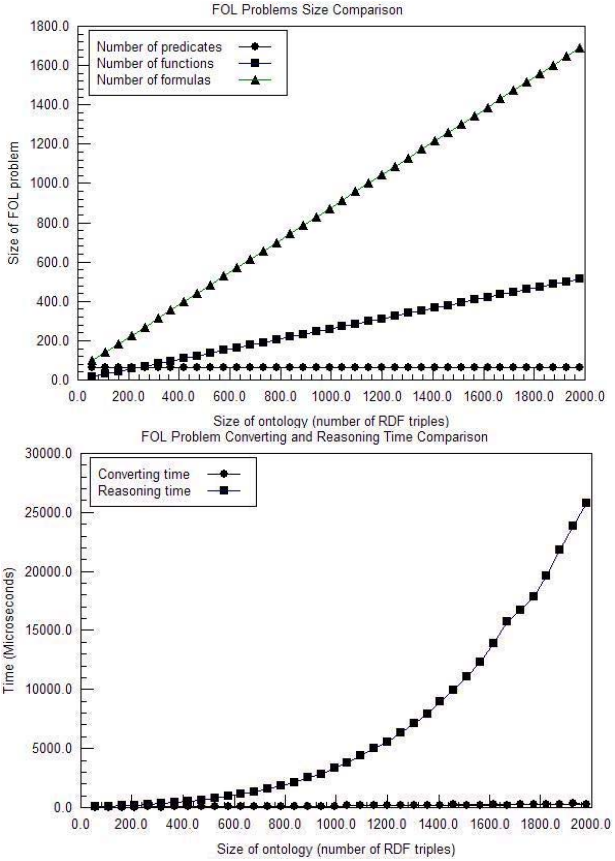

Figure 3. Experimental results of FOL Rule-based Reasoning

rather than the entire specification. Therefore, when OWL specification is converted to FOL problem; only the related portion of the specification needs to be converted so that the reasoning time is reduced. As for time-critical applications, FOL rule-based situation reasoning is not applicable, and specially designed and time-efficient processes need to be used.

7. Conclusions and Future Work

In this paper, a hierarchical OWL-based situation ontology, which satisfies all the requirements in Section 3, is presented. The situation ontology models the core upper ontology for contexts and situations in pervasive computing environments using machine-interpretable semantic-based OWL DL. The hierarchical situation ontology facilitates sharing and reusing of situation information, and can be easily extended with domain specific knowledge. The logic inferences supported by the situation ontology, including the conversion of OWL situation ontology specifications to FOL representations and the performance of FOL rule-based reasoning in terms of problem size and time, are also discussed. We have shown that the situation can be effectively modeled using our OWL-based situation ontology, and that logic reasoning can be performed on the situation ontology specifications.

Future research in this area includes improving situation ontology with temporal logic to reason the values of temporalSituations, and developing algorithms to minimize the FOL problem size converted from situation ontology specifications.

Acknowledgment

The work reported here is supported by the National Science Foundation under the grant number ITR-CYBERTRUST 0430565. The authors would like to thank Dazhi Huang of Arizona State University for many valuable discussions.

References

[1] S. S. Yau, Y. Wang, and F. Karim, "Development of Situation-Aware Application Software for Ubiquitous Computing Environments", *Proc. 26th Ann. Int'l Computer Software and Applications Conf.*, 2002, pp. 233-238.

[2] S. S. Yau, F. Karim, Y. Wang, B. Wang, and S.Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," *IEEE Pervasive Computing, 1(3)*, July-September 2002, pp.33-40.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web", *Scientific American*, May 2001, pp. 34-43.

[4] OWL Web Ontology Language, Available at: http://www.w3.org/TR/owl-ref/.

[5] W3C. Composite Capabilities / Preferences Profile (CC/PP), Available at: http://www.w3.org/Mobile/CCPP/

[6] WAPFORUM. User Agent Profile (UAProf). Available at: http://www.wapforum.org.

[7] Albert Held, Sven Buchholz and Alexander Schill, "Modeling of context information for pervasive computing applications", *Proc. 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI2002)*, 2002.

[8] Indulska, J., Robinson, R., Rakotonirainy, A., and Henricksen, K. "Experiences in Using CC/PP in Context-Aware Systems", *Proc. 4th Int'l Conf. on Mobile Data Management*, 2003, pp. 247-261.

[9] H. Chen, T. Finin, A. Joshi, "Using OWL in a Pervasive Computing Broker", *Proc. Workshop on Ontologies in Open Agent Systems*, 2003, pp. 9-16.

[10] T. Strang, C. Linnhoff-Popien, and K. Frank, "CoOL: A Context Ontology Language to enable Contextual Interoperability", *Proc. 4th IFIP Int'l Conf. on Distributed Applications and Interoperable Systems*, 2003, pp. 236-247.

[11] X. Wang, D. Zhang, T. Gu and H. Pung, "Ontology-Based Context Modeling and Reasoning using OWL", *Proc. 2nd IEEE Ann. Conf' on Pervasive Computing and Communications Workshops*, 2004, pp. 18-22.

[12] Anind K. Dey, "Providing Architectural Support for Building Context-Aware Applications", PhD thesis, College of Computing, Georgia Institute of Technology, 2000.

[13] J. A. Pinto, Temporal Reasoning in the Situation Calculus, PhD Thesis, University of Toronto, 1994.

[14] J. McCarthy., "Situation Calculus with Concurrent Events and Narrative", http://wwwformal.stanford.edu/jmc/ narrative/ narrative.html, 2000.

[15] C. J. Matheus, M. M. Kokar, and K. Baclawski, "A Core Ontology for Situation Awareness", *Proc. 6th Int'l Conf. on Information Fusion, 2003*, pp. 545 –552.

[16] S. S. Yau, D. Huang, H. Gong and H. Davulcu, "Situation-Awareness for Adaptable Service Coordination in Service-based Systems", *Proc. 29th Ann. Int'l Computer Software and Application Conference*, 2005, pp. 107-112.

[17] S. S. Yau, D. Huang, H. Gong, and Y. Yao, "Support for Situation-Awareness in Trustworthy Ubiquitous Computing Application Software", *Jour. of Software Practice and Engineering*, to appear.

[18] V. Haarslev and R. Möller. "Racer: A Core Inference Engine for the Semantic Web", *Proc.2nd Int'l Workshop on Evaluation of Ontology-based Tools,* 2003, pp. 27-36.

[19] Pellet. Pellet - OWL DL Reasoner, 2003. http://www.mindswap.org/2003/pellet.

[20] Using a First Order Logic Prover with OWL. Available at: http://wonderweb.man.ac.uk/owl/first-order.shtml

[21] SWRL: A Semantic Web Rule Language Combining OWL and RuleML. Available at: http://www.w3.org/Submission/SWRL/.

[22] Jena2 Semantic Web Toolkit. Available at: http://www.hpl.hp.com/semweb/jena2.htm.

[23] C. Weidenbach, U. Brahm, T. Hillenbrand, E. Keen, C. Theobalt, and D. Topic, SPASS version 2.0. *Automated Deduction – CADE-18, LNCS 2392*, 2002, pp. 275–279.