Classification-based Situational Reasoning for Task-oriented Mobile Service Recommendation

Marko Luther[1], **Yusuke Fukazawa**[2], **Bertrand Souville**[1], **Kunihiro Fujii**[2]
Takefumi Naganuma[2], **Matthias Wagner**[1], **Shoji Kurakake**[2]

Abstract. We study the case of integrating situational reasoning into a mobile service recommendation system. Since mobile Internet services are rapidly proliferating, finding and using appropriate services requires profound service descriptions. As a consequence, for average mobile users it is nowadays virtually impossible to find the most appropriate service among the many offered. To overcome these difficulties, task navigation systems have been proposed to guide users towards best-fitting services. Our goal is to improve the user experience of such task navigation systems by adding context-awareness (i.e., to optimize service navigation by taking the user's situation into account). In this paper we propose the integration of a situational reasoning engine that applies classification-based inference to context elements, gathered from multiple sources and represented using ontologies. The extended task navigator enables the delivery of situation-aware recommendations in a proactive way. Initial experiments with the extended system indicate a considerable improvement of the navigator's usability.

1 Introduction

Within the growing market for mobile Internet, NTT DoCoMo is today providing services to over 50 million mobile phone subscribers in Japan. The majority of these users enjoy widely diverse contents such as entertainment services (ring-tone downloads, games, etc.), transaction services (money transfer, airline reservation, etc.) and information services (weather forecast, maps and local information, etc.) through DoCoMo's high-speed 3G mobile network. Already today, the number of commercial i-mode sites – DoCoMo's brand of mobile Internet services – ranges in the region of many tenth of thousand. With 4G networks at the horizon that promise still substantially higher bandwidth for data transmissions, the market for services with rich content is expected to expand further.

Key to support such growth is the availability of intelligent service platforms that mediate between services and users by observing the users' activity. These platforms have to assist the user in selecting the most appropriate service from the fast growing service pool to support their real world activities, anytime and anywhere.

Our previously developed task-based service retrieval system for the non-expert mobile user makes it easy to retrieve appropriate services for tackling the users challenges in managing his or her everyday life [25]. The term task refers here to "what the user wants to do" as an expression of the users current activity. Furthermore, the system features a task knowledge base, which contains semantic descriptions of potential activities and links to corresponding services that may be helpful. Although this system enables effective service retrieval, it behaves passive in requiring a users initial input to trigger the problem solving process.

In this paper we propose a proactive extension of our basic system that suggests tasks and services actively, without the need for initial user input. This is achieved by the integration of a situation engine and a situation-based task filter, meant to expose only those tasks that are relevant for a user in a given situation. Taking the user's situation into account avoids the necessity of an initial task query. This leads to a considerable improvement of the navigator's usability, especially for non-expert users who are often not willing to input queries.

The abstract characterization of a user's situation is computed by inference mechanisms on several pieces of context information gathered from multiple context sources [20]. We formulate high-level qualitative context elements in the Web Ontology Language (OWL) [22] and concrete situations as instances within the assertional component (Abox) of a situation ontology. To profit from sound, complete and high-performance classifiers such as FaCT++ [31], Pellet [30] and Racer [12], we restrict ourselves to the OWL DL fragment of OWL. To separate concerns we assume that probabilistic aspects of context representation and reasoning are dealt with at lower representation levels applying bayesian networks or fuzzy logics.

The rest of this paper is organized as follows. After discussing related work in the field of ontology-based context reasoning in the next section, we introduce our task-based service navigator application together with some usage scenarios in Section 3. The overall system architecture that underlies the application is presented in Section 4 and the details on our approach to context representation and classification-based reasoning are given in Section 5. In the closing section we report on our experiences gained from this development.

2 Related Work

Several projects consider the use of ontologies as a key requirement for building context-aware applications. Closely related to our approach is the work done in the CALI project [16] as it explores the use of Description Logics (DL) [1] and the associated inferencing. To overcome the limitations of pure DL-based reasoning, a hybrid approach is proposed. However, our earlier experiments [24] indicate that the suggested loose coupling of a DL reasoner with an external generic rule engine leads to serve performance problems. To achieve completeness both reasoners have to be applied successively until no new facts have been derived. Furthermore, it remains unclear how consistency can be guaranteed taking both the knowledge base and the rule base into account.

[1] DoCoMo Euro-Labs, Landsbergerstr. 312, 80687 Munich, Germany
{luther,souville,wagner}@docomolab-euro.com
[2] NTT DoCoMo Inc., 3-5 Hikari-no-oka, Yokusuka, Kanagawa, 239-8536 Japan
{y-fukazawa,naganuma,kurakake}@netlab.nttdocomo.co.jp

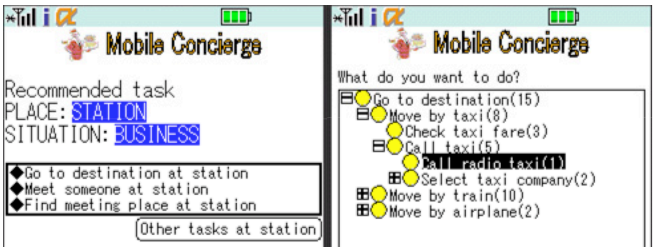

Figure 1. Situation-aware Service Recommender

Other approaches such as CONON [32] and SOUPA/CoBra [4] solely rely on rule-based reasoning which cannot be complete for OWL (not even for OWL Lite [5]) and easily leads to undecidability, as generic rules can be used to simulate role value maps [11].

CONON is an OWL DL encoded upper-context ontology for pervasive computing applications defining almost 200 concepts. Rule-reasoning is used to derive high-level context information and to check its consistency. To cope with the observed delay of several seconds caused by the reasoning process, complex reasoning tasks are computed offline. However, this approach is not feasible in our dynamic setup.

SOUPA, another OWL DL ontology designed for ubiquitous applications, is about the same size as the CONON ontology. Its extension CoBra-Ont is used by a context broker architecture to realize a scenario where people on a university campus come together for a meeting. To limit the reasoning overhead caused by importing standard ontologies, single concepts are mapped to foreign ontology terms. Still, the SOUPA ontology is of a rather high-complexity corresponding $\mathcal{SHOIF}(D)$, because it contains nominals.

An interesting approach to speed up the rule-based inferencing on complex ontologies is to determine relevant contexts required to answer queries using the query-tree method [17]. It remains to be seen how this method extends to our classification-based approach.

3 Situation-aware Service Recommendation

We build on a task-oriented service navigation system [25] that supports the user in finding appropriate services by querying a rich task ontology that represents common sense knowledge about typical complex tasks.

The usage of this basic task navigator is as follows. After having specified a task-oriented query such as "go to theme park" a list of tasks that match this query is sent to the mobile device. Now the user can select the most appropriate task and a corresponding detailed task-model is displayed accordingly. In a final step, associated services can be invoked by establishing an Internet connection to the actual i-mode services.

Figure 1 shows the user interface of the situation-aware variant of the basic service recommender. To explain its functionality, let us assume the following situation.

Situation 1: Important Business Meeting at Tokyo Station
Two travellers, Dawson Campbell and his boss Fiona Davidson, arrive on a Friday morning at the Tokyo main station. Gordon Green, a project partner, is already waiting for them at the platform. The group is looking for a quick transfer to the airport.

Figure 2. Felica Device

To detect the user's location we further assume that the cell phones of Dawson, Fiona and Gordon are equipped with Felica[3] contact-less RFID tags, enabling a two-way communication with Sonys Felica Reader-Writer devices. Whenever a user puts his phone close to a Felica Reader-Writer device (e.g., to make a mobile payment at a train gate) the recommender application retrieves the corresponding location information as a semantic description of this place (cf. Figure 2). Since Sony and NTT DoCoMo just started to deploy their mobile Suica[4] system for JR East at all stations in the Tokyo region, this assumption is not a fiction but reality.

After having passed the gate at Tokyo station, Dawson's phone displays a basic list of tasks, associated with the concept *Station*. This list may include entries such as "Prepare to ride a train", "Buy souvenirs", "Meet someone" etc. While displaying this task-list, Dawson's phone connects to the situational reasoning engine and updates his location to *Tokyo station*.

Before having passed the gate, no tasks are shown on Fiona's phone. Once her location has been detected, a connection to the reasoning engine is established and her current location is updated.

As a result, the situation reasoner infers that Dawson Campbell and Fiona Davidson are travelling together, based on their proximity at the station. In addition, a lookup in the knowledge base reveals that Dawson and Fiona are colleagues and that the scene takes place at a weekdays afternoon.

Because Dawson is located at a *public place* during *office hours* together with *colleagues*, his situation is classified as a *business situation*. His phone shows the inferred situation together with a corresponding list of filtered tasks (shown on the left part of Figure 1). To further specify his needs, Dawson may select one of the recommended tasks ("go to destination" in this case) and finally invoke an associated service (as shown on the right part of Figure 1).

Let us assume another situation taking place at the same location.

Situation 2: Private Meeting at Tokyo Station
Dawson Campbell arrives on a Saturday around noon at the Tokyo main station where Mark Buchanan, his father in law, is awaiting him. They plan to shop for a birthday present for Dawson's wife.

This situation is classified as *private family meeting*, because it takes place during *leisure hours* and only *relatives* are in the proximity. In this case, the situation-aware recommender application suggests tasks that are related to private activities such as "go to movie theater", "go shopping", etc.

The key statement of these scenarios is that task-lists are actually tailored to different situations of the user, even if some context conditions are the same (location in this case). In this respect, our system facilitates users to access the mobile services that fit best to their current situation, purely based on qualitative context information.

[3] <http://www.nttdocomo.co.jp/english/p_s/i/felica>
[4] <http://www.jreast.co.jp/suica/>

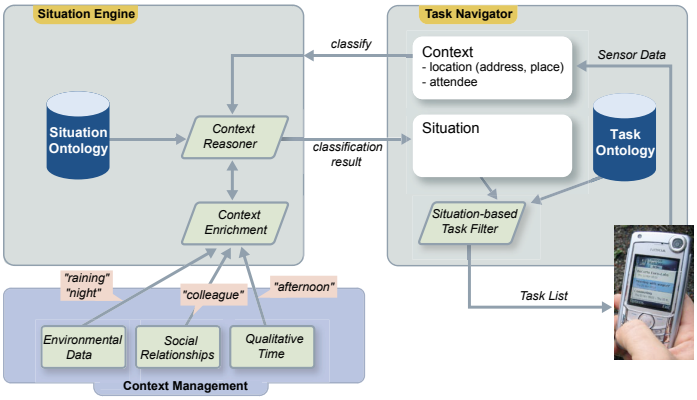

Figure 3. Architecture

4 Architecture

Figure 3 depicts the overall system architecture. The implementation contains two main parts, the situation engine and the task navigator.

The situation engine receives context information that has been collected by the task navigator on the mobile device. Furthermore, this information is enriched by context artifacts, such as environmental data, social relations between companions and a qualitative representation of time, all gathered form a distributed network of context providers. Thereupon, an axiomatized situation instance is constructed and sent to the inference engine. According to the world knowledge encoded in the situation ontology, this instance is classified and the inferred situation is propagated back to the task navigator. A subcomponent of the task navigator, the task filter, detects the most appropriate task nodes within the task ontology by matching the derived situation with the task-specific categories. Finally, a representation of the resulting task list is constructed by the task navigator and presented to the user on his mobile device for further navigation and service selections.

The task ontology stores descriptions for abstract as well as concrete tasks and their interrelations as semantic descriptions. Large and abstract tasks are thereby described by sequences of smaller subtasks. In addition, abstract tasks are annotated with enabling context conditions and concrete tasks are linked to appropriate information services via Uniform Resource Identifiers. The task structures are defined in terms of the process model of the OWL-S ontology [21]. Each task node is represented as a service class and categorized according to the high-level context concepts such as *Business_meeting*, defined within the situation ontology. The context conditions describing the applicability of a task node are thereby encoded as corresponding OWL-S service profiles. More details about our task ontology can be found elsewhere [26].

5 Context Representation and Classification

We adopted the IST MobiLife[5] Context Management Framework [7] to achieve interoperability between context sources from diverse domains by defining an XML-based context meta model. The elements of this meta model are linked to ontologies that define the basic contextual categories, used to represent qualitative aspects of context information.

[5] `http:\\www.ist-mobilife.org`

We refer to an ontology as a logical theory accounting for the intended meaning of a formal vocabulary, i.e. its ontological commitment to a particular conceptualization. Therefore, the decidability of the selected ontology language is crucial. The OWL DL fragment of the OWL fulfills this requirement, is highly expressive and has the potential to become the standard ontology language for the Semantic Web. Its selection as the ontology language of choice resulted in the construction of high-quality ontologies (i.e., ontologies that are proven consistent by fully automatic inference engines that are available for OWL DL). It is important to note that we do not propose the ontologies described hereafter as the main representation format for all aspects of context modeling, as ontologies are limited to the formulation of qualitative aspects and the available inference engines are generally weak in handing large amounts of data efficiently.

The context ontologies are composed of eight interrelated components defining more than 300 concepts, 200 properties and 300 individuals. They provide a general vocabulary for temporal and spatial concepts, agents as well as devices. Being informed by the vCard standard, the iCalendar representation and the FOAF (Friend-of-a-friend) format, an extension for the precise modeling of complex social relations has been developed. All component ontologies are integrated by a situation ontology that defines a top-level concept named *Situation* (cf. Figure 4). This concept is refined by concepts such as *Private* and *Business* by referring to concepts and relations defined in the component ontologies.

We exemplarily sketch the OWL definitions of two typical situations using standard DL syntax [1]. A person's situation is classified as *Business*, if he is either located at a business place (such as an office) or at a public place (e.g., a train station) during office hours.

$$\begin{aligned} \mathit{Business} \;:=\; & \mathit{Situation} \sqcap (\exists\, \mathit{location} \,.\, \mathit{Business\_place} \sqcup \\ & (\exists\, \mathit{location} \,.\, \mathit{Public\_place} \sqcap \exists\, \mathit{time} \,.\, \mathit{Office\_hour})) \end{aligned}$$

A person is participating a family meeting if he or she is in a private meeting situation where all participants are relatives.

$$\mathit{Family\_meeting} \;:=\; \mathit{Situation} \sqcap (\forall\, \mathit{company} \,.\, \mathit{Relative})$$

Situational reasoning is realized using a DL reasoning engine that classifies concrete individual situations w.r.t. the ontology. Let us consider the Situation 1 introduced in Section 3. First, each piece of context information such as the location (Tokyo station), the time (Sunday morning), and all companions (Dawson's boss Fiona and his project partner Gordon) are represented in terms of vocabulary formalized by the context ontologies. This requires the mapping of sensed quantitative data to qualitative representations (e.g. a timestamp is mapped to an individual in the Abox representing a Friday morning). The qualitative representations are enriched by the world-knowledge formalized in the component ontologies and are combined to an Abox individual in the situation ontology.

Computed by the reasoning engine, the direct concept type for the situation instance according to Scenario 1 is *Important_meeting*. In this case, the location of the scene is a public place (as *tokyo_station* is an instance of the concept *Station*, which in turn is a subconcept of *Public_place*) during office hours (as the individual *friday_morning* is classified as *Office_hours*) and the main actor Dawson is accompanied by his supervisor and a business partner. Similarly, the situation instance constructed for Scenario 2 is classified as *Family_meeting* as it takes place at a public location during leisure time and only relatives are detected in the proximity of Dawson.

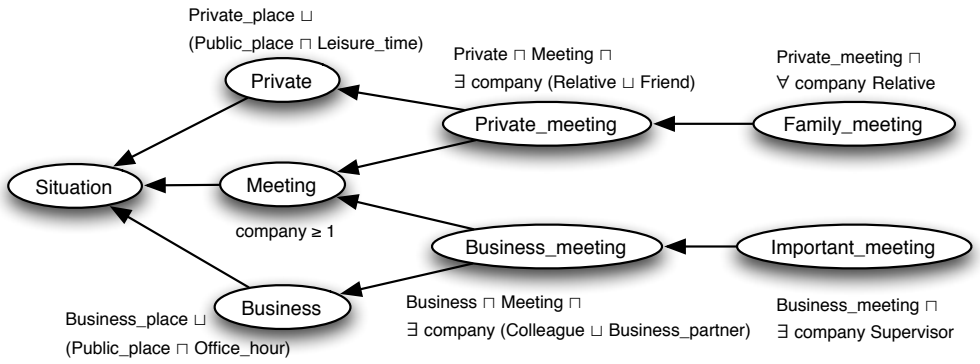

Figure 4. Situation Ontology Fragment

The situational reasoning process described above is supported by deductions in all component ontologies. For example, the agent ontology specifies in detail the semantics of social relations between people. Based on the knowledge encoded within the ontology, it can be inferred that two persons (like Dawson and Fiona) are colleagues, taking into account the transitivity of this relationship in case they have a common colleague. Similarly, even if no direct relation between Dawson and Mark is specified it can be inferred that Mark is Dawson's father in law (defined to be the father of the spouse of a person), because Dawson's wife Madeleine is known to be the child of Mark. In this case, the subproperty and inverse property specifications within the agent ontology enable this logical inference: *wife* is defined as a subproperty of *spouse* and *father* is the inverse of *child*.

6 Discussion

We integrated a situational reasoning engine into a real-world mobile service application. Our classification-based approach relies on ontology technology for the representation and reasoning on context information. As the scalable management of data is not a core feature of pure ontology-based context management and typical context models are usually rather large, we restricted its scope to high-level qualitative context elements. Lower-level context information is represented according to an XML-based meta model and managed separately. The arising reasoning problems are answered by a Description Logic (DL) [1] inference engine that provides complete reasoning support for the decidable fragment of OWL.

The use of the standard representation language OWL and the standardized reasoner interface DIG [2] (a stateless HTTP-based protocol with XML syntax) enabled us to directly compare the influence of different context ontologies and reasoners on the overall system performance. We observed that the inference technology as implemented in modern DL reasoners made significant progress during the last years. Novel optimization techniques enabled a tremendous increase in performance, and also the coverage was greatly extended. By now most systems can be accessed via DIG, and support nominals as well as Abox reasoning directly. FaCT++ and Pellet support $\mathcal{SHOIQ}(\mathcal{D})$ (OWL DL extended by qualified cardinality restrictions) and RacerPro supports $\mathcal{SHIQ}$ including approximated nominals and reasoning with concrete domains.

Nevertheless we observed several limitations in the available technology (see [18] for details). The import mechanism of OWL, which brings all triples into the importing ontology, has a limited use for the sharing and reuse of ontologies. An appropriate mechanism on the syntactic as well as the semantic level is necessary for referencing entities in another ontology without inheriting all of its complexity. Furthermore, our modeling of context ontologies would benefit from additional constructs such as qualified cardinality restrictions and a richer object property structure that would allow the specification of reflexive, irreflexive, symmetric and anti-symmetric properties as well as property chains and disjoint property axioms. Reasoning support for the DL-safe fragment [23] of SWRL [14] and for concrete domains on user defined datatypes would allow us to further enhance the quality of our situation engine. While concrete domain reasoning and support for SWRL is already available in some inference engines, and most of the requested additional language constructs are part of the OWL 1.1 draft[6] created by the ad-hoc OWL community, an improved import mechanisms as given by the $\mathcal{E}$-connection mechanism [10] and implemented in Pellet is not included.

At first, we experimented with the DIG interface to realize the communication between our application and the inference engine. However, DIG 1.1 does not support the removal of specific axioms making it necessary to re-submit the complete ontology for each request to our situation engine. This is especially awkward for our application where only a very small part of the assertional knowledge changes between two requests. As active members of the informal DIG 2.0 working group[7] we therefore propose a modular extension to the interface that supports incremental reasoning and retraction. Unfortunately, current reasoner typically only provide some kind of batch-oriented reasoning procedure. A notable exception is Racer which offers low-level retraction support for most of its statements. Still, because of the lack of algorithms for appropriately handling incremental additions as well as retractions, Racer initiates a complete reclassification after each change in the ontology. Initial empirical results, performed with an experimental version of Pellet, indicate that incremental classification algorithms for $\mathcal{SHOIN}(\mathcal{D})$ can be quite effective [28].

The ability to handle simultaneous requests is one of the key requirements in our dynamic mobile setting. However, current inference engines do not implement any transaction management. Only for Racer, support for dispatching, load balancing and caching of OWL-QL [6] queries is available via the RacerManager [8]. As OWL-QL does not support modifications of an ontology, we had to implement our own transaction management system that enables the sharing of reasoning resources between requests, but avoids the necessity to maintain a separate knowledge base for each user.

[6] ⟨http://www-db.research.bell-labs.com/user/pfps/owl⟩
[7] ⟨http://homepages.cs.manchester.ac.uk/~seanb/dig⟩

It has been observed before [17][32] that the delay caused by ontology-based inferencing easily becomes a major obstacle for realistic applications. This is especially problematic for ontologies that constantly change, because well-established optimization techniques such as tabling (used in various rule-based inference engine) cannot be applied. As a consequence of the high worst-case complexity of expressive DLs, such as $\mathcal{SHOIN}(\mathcal{D})$ underlying OWL DL, modern DL reasoners implement a suite of optimization techniques to achieve acceptable performance. The efficiency of implementations on concrete cases depends therefore on the applicability of optimizations, which varies with the language features in use. For example, the use of domain and range restrictions can lead to cycles in a Tbox for which termination of the tableaux algorithm can only be ensured by blocking. However, known blocking strategies for $\mathcal{SHOIN}$ are less effective if inverse roles are involved. On the other hand, if nominals do not occur in an ontology blocking can be realized more efficiently [13]. Therefore we avoid the use of standard ontologies, such as the $\mathcal{SHOIF}(\mathcal{D})$ entry sub-ontology of time [27]. It has to be seen how the recently suggested techniques for optimizing DL reasoning in the presence of nominals [29] perform in practice.

We optimized our initial ontologies by removing nominals and most of the domain and range restrictions. Furthermore, we reduced the number of loaded axioms and objects (especially Abox individuals) and axioms by splitting the ontology in small components and by separating ontologies in A- and Tboxes to cope with the limits of the OWL import statement. This step resulted in a performance gain of up to 1,5 seconds per request. Furthermore, we compared different retraction strategies using Racer. The simplest form of retraction is reloading of ontologies and can be accelerated by either loading from a pre-classified image or by cloning an ontology in memory. For small Aboxes cloning outperformed true retraction realized with forgot statements. However, the strategy performed best was to keep situation individuals up to a certain number (about 20 in our case) in the Abox before cloning a fresh pre-loaded Abox. Of course, keeping individuals and axioms in the Abox is only possible if they do not influence later classifications.

The time to compute our comparable simple reasoning problems is dominated by the communication overhead caused by the reasoner interface. Accessing Racer via its native API using TCP is about 1,5 times faster then the access via HTTP/DIG and even 2 times faster then the access realized with the triple-oriented framework Jena2 [3]. Naturally, we achieved the best performance by using the Pellet reasoner running in the same Java virtual machine and this way completely avoiding any external communication.

Because existing performance results of DL reasoners are often limited to static Tbox classification, we plan to perform a detailed analyze of the influence of different retraction strategies for dynamic assertional reasoning, to compare the performance of interfaces and to test the effect of the ontology size and complexity on realistic reasoning tasks. By that we hope to gain inside on how to further optimize our situation engine.

Our current prototype has only a limited support for automatic context acquisition. We plan to advance the prototype towards the use of more actual context information from the real world. Planed extensions will combine GPS-based location information with the RFID-based context tags we use currently for location tracking, as well as or short distance wireless communication technologies such as Bluetooth to detect people in proximity [19].

REFERENCES

[1] F. Baader et al., *The Description Logic Handbook*, Cambridge University Press, Cambridge, January 2003.

[2] S. Bechhofer, R. Möller, and P. Crowther, 'The DIG Description Logic Interface', in *Proc. of the Int. Workshop on Description Logics*, (2003).

[3] J. Carrol et al., 'Jena: Implementing the Semantic Web recommendations', Technical Report HPL-2004-146, HP Labs, (2004).

[4] H. Chen, T. Finin, and A. Joshi, 'The SOUPA ontology for pervasive computing', in *Ontologies for Agents*, eds., V. Tamma, S. Cranefield, and T. Finin, Springer, (July 2005).

[5] K. de Bruin and D. Fensel, 'Owl-', WSML Deliverable D20.1, (2005).

[6] R. Fikes et al., 'OWL-QL', *Journal of Web Semantics*, (2005).

[7] P. Floréen, M. Przybilski, P. Nurmi, J. Koolwaaij, A. Tarlano, M. Wagner, and M. Luther et al., 'Towards a context management framework for MobiLife', in *Proc. of the IST Summit*, (June 2005).

[8] J. Galinski et al., 'Development of a server to support the formal Semantic Web query language OWL-QL', In Horrocks et al. [15].

[9] Y. Gil, E. Motta, R. Benjamins, and M. Musen, eds. *Proc. of the 4th Int. Semantic Web Conference*, volume 3729 of *LNCS*. Springer, 2005.

[10] B. Grau, B. Parsia, and E. Sirin, 'Combining OWL ontologies using $\mathcal{E}$-connections', *Journal of Web Semantics*, **4**(1), (2005).

[11] B. Grosof, I. Horrocks, R. Volz, and S. Decker, 'Combining logic programs with Description Logic', in *Proc. of the Int. WWW Conf.*, (2003).

[12] V. Haarslev and R. Möller, 'Racer: A core inference engine for the Semantic Web Ontology Language (OWL)', in *Proc. of the 2nd Int. Workshop on Evaluation of Ontology-based Tools*, pp. 27–36, (2003).

[13] V. Haarslev, R. Möller, and M. Wessel, 'Description Logic inference technology: Lessions learned in the trenches', In Horrocks et al. [15].

[14] I. Horrocks and P Patel-Schneider, 'A proposal for an OWL rules language', in *Proc. of the Int. WWW Conf.*, pp. 723–731. ACM, (2004).

[15] I. Horrocks et al., ed. *Int. Workshop on Description Logics*, July 2005.

[16] D. Khushraj and O. Lassila, 'CALI: context awareness via logical inference', in *Proc. of the Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications*, (November 2004).

[17] J. Lee, I. Park, and S. Hyun, 'Application-oriented context pre-fetch method for improving inference performance in ontology-based context management', in *Workshop on Contexts and Ontologies*, (2005).

[18] Th. Liebig, M. Luther, O. Noppens, M. Paolucci, M. Wagner, and F.. von Henke, 'Building applications and tools for OWL', in *Proc. of the OWL Experiences and Directions WS*, (November 2005).

[19] M. Luther, S. Böhm, M. Wagner, and J. Koolwaaij, 'Enhanced presence tracking for mobile applications', In Gil et al. [9].

[20] M. Luther, B. Mrohs, M. Wagner, S. Steglich, and W. Kellerer, 'Situational reasoning – a practical OWL use case', in *Proc. of the 7th Int. Symp. on Autonomous Decentralized Systems*, (April 2005).

[21] D. Martin et al., 'OWL-S: Semantic Markup for Web Services', W3C Member Submission, The OWL Serivces Coalition, (November 2004).

[22] D. McGuinness and F. van Harmelen, 'OWL Web Ontology Language overview', W3C Recommendation, (February 2004).

[23] B. Motik, U. Sattler, and R. Struder, 'Query answering for OWL-DL with rules', in *Proc. of the Int. Semantic Web Conf.*, (2004).

[24] B. Mrohs, M. Luther, R. Vaidya, and M. Wagner et al., 'OWL-SF – a distributed semantic service framework', in *Proc. of the Workshop on Context Awareness for Proactive Systems*, pp. 67–77, (June 2005).

[25] T. Naganuma and S. Kurakake, 'Task knowledge based retrieval for service relevant to mobile user activity', In Gil et al. [9], pp. 959–973.

[26] T. Naganuma and M. Luther et al., 'Task-oriented mobile service recommendation enhanced by a situational reasoning engine', in *Proc. of the 1st Asian Semantic Web Conference (ASWC'06)*, (2006). To Appear.

[27] F. Pan and J. Hobbs, 'Time in OWL-S', in *Proceedings of the AAAI Spring Symposium on Semantic Web Services*, (2004).

[28] B. Parsia et al., 'Towards incremental reasoning through updates in OWL-DL', in *Reasoning on the Web WS*, (2006). To Appear.

[29] E. Sirin, B. C. Grau, and B. Parsia, 'From wine to water: Optimizing Description Logic reasoning for nominals', in *Int. Conf. on the Principles of Knowledge Representation and Reasoning*, (2006). To Appear.

[30] E. Sirin and B. Parsia, 'Pellet: An OWL DL reasoner', in *Proc. of the Int. Workshop on Description Logics*, pp. 212–213, (2004).

[31] D. Tsarkov and I. Horrocks, 'Ordering heuristics for Description Logic reasoning', in *Proc. of the 19th Int. Conf. on AI (IJCAI'05)*, (2005).

[32] X. Wang, Q. Zhang, T. Gu, and H. K. Pung, 'Ontology-based context modeling and reasoning using OWL', in *Proc. of the Workshop on Context Modeling and Reasoning (PerCom'04)*, pp. 18–22, (March 2004).