A mobile matchmaker for the Ubiquitous Semantic Web

Floriano Scioscia, Michele Ruta, Giuseppe Loseto, Filippo Gramegna, Saverio Ieva,
Agnese Pinto, Eugenio Di Sciascio
Politecnico di Bari, via E. Orabona 4, I-70125 Bari, Italy
{floriano.scioscia, michele.ruta, giuseppe.loseto, filippo.gramegna, saverio.ieva,
agnese.pinto, eugenio.disciascio}@poliba.it

The Semantic Web and Internet of Things visions are converging toward the so-called Semantic Web of Things (SWoT). It aims to enable smart semantic-enabled applications and services in ubiquitous contexts. Due to architectural and performance issues, it is currently impractical to use existing Semantic Web reasoners. They are resource consuming and are basically optimized for standard inference tasks on large ontologies. On the contrary, SWoT use cases generally require quick decision support through semantic matchmaking in resource-constrained environments. This paper presents Mini-ME, a novel mobile inference engine designed from the ground up for the SWoT. It supports Semantic Web technologies and implements both standard (subsumption, satisfiability, classification) and non-standard (abduction, contraction, covering) inference services for moderately expressive knowledge bases. In addition to an architectural and functional description, usage scenarios are presented and an experimental performance evaluation is provided both on a PC testbed (against other popular Semantic Web reasoners) and on a smartphone.

Introduction

Semantic Web technologies have been acknowledged as tools to promote interoperability and intelligent information processing in ubiquitous computing. Scenarios include supply chain management, u-commerce (Liu, 2013; Ruta, Di Noia, Di Sciascio, Piscitelli, & Scioscia, 2007), peer-to-peer resource discovery (Ruta, Di Sciascio, & Scioscia, 2011) and so on. The increasing computational resources and communications effectiveness of mobile devices enable ubiquitous processing and exchange of rich and structured information for context-aware resource discovery and decision support. The Semantic Web and the Internet of Things paradigms are converging more and more toward the so–called Semantic Web of Things (SWoT) (Scioscia & Ruta, 2009; Pfisterer et al., 2011). It enables semantic-enhanced pervasive computing by embedding intelligence into ordinary objects and environments through a large number of heterogeneous micro-devices, each conveying a small amount of information.

Such a vision requires an increased flexibility and autonomy of ubiquitous knowledge-based systems in information encoding, management, dissemination and discovery. User agents running on mobile personal devices should be able to discover dynamically the best available resources according to user's profile and preferences, in order to support her current tasks through unobtrusive and context-dependent suggestions. Reasoning and query answering are particularly critical issues, stimulating the need for further specialized inference services in addition to classical ones like *subsumption* and *satisfiability* check. Furthermore, mobile computing platforms (*e.g.*, smartphones, tablets) are still constrained by hardware/software limitations with respect to typical setups for Semantic Web reasoning engines. In fact, architectural and performance issues affect the porting of current OWL-based reasoners, designed for the Semantic Web, to mobile devices (Yus, Bobed, Esteban, Bobillo, & Mena, 2013).

This paper presents *Mini-ME (the Mini Matchmaking Engine)*[1], a compact matchmaker and reasoner for the $\mathcal{ALN}$ (Attributive Language with unqualified Number restrictions) Description Logic (DL). It is aimed to semantic matchmaking for resource/service discovery in mobile and ubiquitous contexts, although it is also a general-purpose Semantic Web inference engine. The reduced expressivity of the logical language is compensated by an increased mobility level and provided quality of the resource discovery. Optimized non-standard inference services allow a fine-grained categorization and ranking of matching resources w.r.t. a request, providing both a distance metric and a logic-based explanation of the outcomes. Mini-ME is suitable to a widespread class of applications where a large number of low-complexity component resources can be aggregated to build composed services with growing semantic complexity. This is fit for the computational and power supply limitations of resource providers in ubiquitous contexts and to their short storage availability. An "agile" service discovery architectures able to select, assemble and orchestrate *on the fly* many elementary components is more manageable and effective in ubiquitous applications.

Mini-ME uses the OWL API (Horridge & Bechhofer, 2009) to parse and manipulate Knowledge Bases in all OWL 2[2] supported syntaxes. It exploits structural inference algorithms on unfolded and CNF (Conjunctive Normal Form) normalized concept expressions for efficient computations also on resource-constrained platforms. Mini-ME implements both standard reasoning tasks for Knowledge Base (KB) management (subsumption, classification, satisfiability) and non-standard inference services for semantic-based resource discovery and ranking (abduction, contraction (Colucci et al., 2007), covering (Ragone et al., 2007)). It is developed in Java, with *Android* as the main target computing platform. Mini-ME supports the *OWLlink* protocol (Liebig, Luther, Noppens, & Wessel, 2011) for standard application-reasoner interaction. Furthermore, reasoner and GUI (Graphical User Interface) plug-ins have been developed for the *Protégé*[3] ontology editor (Gennari et al., 2003). Mini-ME has already been employed in prototypical testbeds on mobile and embedded devices for ubiquitous and pervasive computing scenarios.

The remainder of the paper is organized as follows. The next section highlights relevant related work. A functional and architectural description of the system is given, followed by two usage scenarios, respectively in ontology engineering with Protégé and in mobile semantic augmented reality. Experimental evaluation results are provided for both standard and non-standard inferences, in mobile as well as PC settings, before closing remarks.

Related work

When processing semantic-based information to infer entailed implicit knowledge, painstaking optimization is needed to achieve acceptable performance for adequately expressive languages (Baader, Hollunder, Nebel, Profitlich, & Franconi, 1994; Horrocks & Patel-Schneider, 1999). This is specifically true in case of logic-based matchmaking for mobile computing, which is characterized by resource limitations affecting not only processing, memory and storage capabilities, but also energy consumption. Most mobile inference engines currently provide only rule processing for entailments materialization in a KB (Koch, Meyer, Dignum, & Rahwan, 2006; Tai, Keeney, & O'Sullivan, 2011; Kim, Park, Hyun, & Lee, 2010; Motik, Horrocks, & Kim, 2012), so resulting unsuitable to support applications requiring non-standard inference tasks and extensive reasoning over ontologies (Motik et al., 2012). More expressive languages could be used by adapting tableaux algorithms –whose variants are implemented in reasoners running on PCs– to mobile computing platforms, but an efficient implementation of reasoning services is still an open problem. Several techniques (Horrocks & Patel-Schneider, 1999) allow to increase expressiveness or decrease running time at the expense of main memory usage, which is precisely the most constrained resource in mobile systems.

Focusing on ubiquitous contexts, semantic-based resource discovery was early investigated in (Avancha, Joshi, & Finin, 2002). There, the need for discovery mechanisms more powerful than string-matching was clearly pointed out for the first time. The issue of approximate matches lacking exact ones was discussed, but no formal frameworks were given. In (von Hessling, Kleemann, & Sinner, 2004) semantic user profiles were introduced to increase accuracy in matching services in a mobile environment. If there was no intersection between user interests and service offers, authors concluded the user was not interested in the service; a complete and integrated solution for matching degree calculation was not provided. *Pocket KRHyper* (Sinner & Kleemann, 2005) was the first reasoning engine specifically designed for mobile devices. It supported the $\mathcal{ALCHIR}$+ DL and was built as a Java ME (Micro Edition) library. Pocket KRHyper was exploited in a DL-based matchmaking framework between user profiles and descriptions of mobile resources/services (Kleemann & Sinner, 2005). However, frequent "out of memory" errors strongly limited the size and complexity of manageable logic expressions. To overcome performance constraints, tableaux optimizations to reduce memory consumption were introduced in (Steller & Krishnaswamy, 2008) and implemented in *mTableau*, a modified version of Java SE *Pellet* reasoner (Sirin, Parsia, Cuenca Grau, Kalyanpur, & Katz, 2007). Comparative performance tests were executed on a PC, showing faster turnaround times than both unmodified Pellet and *Racer* (Haarslev & Müller, 2001) reasoner. Nevertheless, the Java SE technology is not expressly tailored to the current generation of handheld devices. In fact, other relevant inference engines cannot run on common mobile platforms, since they rely on Java class libraries incompatible with most widespread mobile OS (*e.g.*, Android). In (Yus et al., 2013) four Semantic Web reasoners were successfully ported to the Android platform (Pellet, *CB* (Kazakov, 2009), *Hermit* (Shearer, Motik, & Horrocks, 2008) and *JFact*, a Java port of *Fact++* (Tsarkov & Horrocks, 2006)), albeit with significant rewriting or restructuring effort in some cases. Similarly, in (Kazakov & Klinov, 2013) the *ELK* reasoner was optimized and evaluated on Android.

Nevertheless, all ported systems were designed mainly for batch jobs over large ontologies and/or expressive languages. This makes mobile device usage less suitable due to computation and memory constraints. The non-standard services of Mini-ME are more useful in ubiquitous scenarios, where mobile agents must provide quick decision support and/or on-the-fly organization in intrinsically unpredictable environments like the ubiquitous ones. Moreover, it can be observed that the above systems, like the matchmaking framework proposed in (Kleemann & Sinner, 2005), only support standard inference services such as satisfiability and subsumption, which provide only binary "yes/no" answers.

Consequently, they can only distinguish among *full* (*subsume*), *potential* (*intersection-satisfiable*) and *partial* (*disjoint*) match types, as defined in (Colucci et al., 2007) and (Li & Horrocks, 2004) respectively. Analogously, in the HTTP-based ubiquitous infrastructure by (Vazquez & López-de Ipiña, 2007), queries allow only exact matches with facts derived from a support knowledge base. Non-standard inferences like abduction and contraction are needed to support approximate matches, semantic ranking and explanations of outcomes (Colucci et al., 2007).

In latest years, the bad worst-case complexity of OWL language stimulated a different approach to implement reasoning tools. It was based on simplifying both the underlying logic languages and admitted KB axioms, so that structural algorithms could be adopted, while maintaining expressiveness enough for broad application areas. In (Baader, Brandt, & Lutz, 2005), the basic $\mathcal{EL}$ DL was extended to $\mathcal{EL}^{++}$, a language deemed suitable for various applications, characterized by very large ontologies with moderate expressiveness. A structural classification algorithm was also devised, which allowed high-performance $\mathcal{EL}^{++}$ ontology classifiers such as *CEL* (Baader, Lutz, & Suntisrivaraporn, 2006), *Snorocket* (Lawley & Bousquet, 2010) and ELK (Kazakov, Krötzsch, & Simančík, 2014). OWL 2 profiles definition complies with this perspective, focusing on language subsets of practical interest for important application areas rather than on fragments with significant theoretical properties. In a parallel effort motivated by similar principles, in (Ruta, Di Noia, Di Sciascio, Piscitelli, & Scioscia, 2008) an early approach was proposed to adapt non-standard logic-based inferences to pervasive computing contexts. By limiting expressiveness to the $\mathcal{AL}$ language, acyclic, structural algorithms were adopted reducing standard (*e.g.*, subsumption) and non-standard (*e.g.*, abduction and contraction) inference tasks to set-based operations (Di Noia, Di Sciascio, & Donini, 2007). KB management and reasoning were then executed through a data storage layer, based on a mobile RDBMS (Relational DBMS). Such an approach was further investigated in (Ruta, Scioscia, Di Noia, & Di Sciascio, 2009) and (Ruta et al., 2011), by increasing the expressiveness to $\mathcal{ALN}$ DL and allowing larger ontologies and more complex descriptions, through the adoption of both mobile OODBMS (Object-Oriented DBMS) and performance-optimized data structures. Finally, in (Ruta, Scioscia, & Di Sciascio, 2010) expressiveness was extended to $\mathcal{ALN}$(D) DL with fuzzy operators. The above tools were designed to run on Java ME devices and were adopted in several case studies in ubiquitous computing, employing semantic matchmaking over moderately expressive KBs. The reasoning engine presented here recalls lessons learned in those previous efforts, and aims to provide a standards-compliant implementation of most common inferences (both standard and non-standard) for the most widespread mobile platform. In recent years, several case studies and prototypes have been developed for different ubiquitous scenarios to prove the feasibility and benefits of non-standard inferences, including ubiquitous commerce (Di Noia et al., 2008), ambient intelligence and infomobility services (Ruta, Scioscia, Di Noia, & Di Sciascio, 2010), home and building automation (Ruta, Scioscia, Loseto, & Di Sciascio, 2014).

System outline

A description of the proposed matchmaker is provided hereafter: a short recall on the supported logic language is followed by an overview of the included inference services and finally by architectural and implementation details.

Supported language

In DL-based reasoners, an ontology $\mathcal{T}$ (a.k.a. Terminological Box or TBox) is composed by a set of assertions in the form $A \sqsubseteq D$ (inclusion) or $A \equiv D$ (definition), where A and D are concept expressions. Particularly, a *simple-TBox* is an acyclic TBox such that: (i) A is always an atomic concept; (ii) if A appears in the left hand side (lhs) of a concept definition assertion, then it cannot appear also in the lhs of any concept inclusion assertion. Mini-ME supports the $\mathcal{ALN}$ (Attributive Language with unqualified Number restrictions) DL, which has polynomial computational complexity for standard and non-standard inferences in simple-TBoxes, whose depth of concept taxonomy is bounded by the logarithm of the number of axioms in it (see (Di Noia et al., 2007) for further explanation). Actually, such DL fragment has been selected because it grants good worst-case complexity and memory efficiency of non-standard inference algorithms for semantic matchmaking. Syntax and semantics of $\mathcal{ALN}$ DL constructs are summarized in Table 1.

According to the W3C (World Wide Web Consortium) Recommendation issued by the OWL Working Group for the OWL 2 ontology language, an OWL 2 ontology is basically an RDF (Resource Description Framework)[4] graph referring to the OWL 2 vocabulary[5], serialized in RDF/XML syntax or in one of the other optional syntaxes. As part of the activity of the W3C, *OWLlink* (Liebig et al., 2011) has been defined as a new interface to allow communication between applications and OWL 2 reasoners. It is a functional *Tell/Ask* (Levesque, 1984) interface exploiting HTTP as the underlying transfer protocol. There is a close correspondence between OWL 2 and OWLlink syntax.

Inference services

When loading a KB, Mini-ME performs a preprocessing in order to execute *unfolding* and *CNF normalization*. Particularly, given a TBox $\mathcal{T}$ and a concept C, the **unfolding** procedure recursively expands references to axioms in $\mathcal{T}$ within the concept expression itself. In this way, $\mathcal{T}$ is not needed

Table 1
Syntax and semantics of $\mathcal{ALN}$ constructs and simple-TBoxes

Name	Syntax	Semantics
Top	$\top$	Δ^I
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^I \cap D^I$
Atomic negation	$\neg A$	$\Delta^I \backslash A^I$
Universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^I \rightarrow d_2 \in C^I\}$
Number restriction	$\geq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^I\} \geq n\}$
	$\leq nR$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^I\} \leq n\}$
Inclusion	$A \sqsubseteq D$	$A^I \subseteq D^I$
Equivalence	$A \equiv D$	$A^I = D^I$

any more when executing subsequent inferences. **Normalization** transforms the unfolded concept expression in CNF through a set of pre-defined substitutions (Ruta et al., 2011). Any $\mathcal{ALN}$ concept expression C can be reduced in CNF as: $C \equiv C_{CN} \sqcap C_{LT} \sqcap C_{GT} \sqcap C_{\forall}$, where C_{CN} is the conjunction of (possibly negated) atomic concept names, C_{LT} (respectively C_{GT}) is the conjunction of $\leq$ (resp. $\geq$) number restrictions (no more than one per role), and $C_{\forall}$ is the conjunction of universal quantifiers (no more than one per role; fillers are recursively in CNF). Normalization preserves semantic equivalence w.r.t. models induced by the TBox; furthermore, CNF is unique up to commutativity of conjunction operator (Di Noia et al., 2007). The normal form of an unsatisfiable concept is simply $\bot$.

Mini-ME was devised as a semantic matchmaker, *i.e.*, a tool to find the best resources for a given request, when both resource and request descriptions are satisfiable concept expressions w.r.t. a common ontology. Mini-ME exploits structural algorithms for standard and non-standard inference services on (unfolded and normalized) concept expressions. Main peculiarity is a careful optimization of the implementation of both algorithms and data structures, which enable efficient computations even on resource-constrained devices such as mobile and embedded ones.

The following standard reasoning services are currently supported:

- **Concept Satisfiability** (a.k.a. consistency). In a semantic matchmaking framework, given a request D and a supplied resource S as concept expressions w.r.t. a common TBox $\mathcal{T}$, satisfiability allows to determine whether there is a partial (disjoint) match or not, by checking whether $\mathcal{T} \models S \sqcap D \sqsubseteq \bot$ holds or not. Due to CNF properties, satisfiability check is trivially performed during normalization.
- **Subsumption check**. Subsumption determines whether the resource is a full (subsume) match for the request or not, by checking whether $\mathcal{T} \models S \sqsubseteq D$ holds or not. The classic structural subsumption algorithm is exploited, reducing the procedure to a set containment test (Baader, Calvanese, Mc Guinness, Nardi, & Patel-Schneider, 2002).

Furthermore, three non-standard inference services were also implemented, allowing to (i) provide explanation of outcomes beyond the trivial "yes/no" answer of satisfiability and subsumption tests, (ii) enable a logic-based relevance ranking of a set of available resources w.r.t. a specific query (Ruta et al., 2011) and aggregate resources in order to satisfy complex requests:

- **Concept Contraction** (Colucci et al., 2007): given a request D and a supplied resource S, if they are not compatible with each other, Contraction determines which part of D is conflicting with S. As shown in the flowchart in Figure 1, if one retracts conflicting requirements in D, G (for *Give up*), a concept K (for *Keep*) is obtained, representing a contracted version of the original request, such that $K \sqcap S$ is satisfiable w.r.t. $\mathcal{T}$. The solution G to Contraction represents "why" $D \sqcap S$ are not compatible. The algorithm in Figure 1 is recursive.
- **Concept Abduction** (Colucci et al., 2007): whenever D and S are compatible, but S does not imply D, Abduction allows to determine what should be hypothesized in S in order to completely satisfy D also enabling a logic-based relevance ranking of a resource w.r.t. a given request (Ruta et al., 2011). The solution H (for *Hypothesis*) to Abduction represents "why" the subsumption relation $\mathcal{T} \models S \sqsubseteq D$ does not hold. H can be interpreted as *what is requested in D and not specified in S*. Also in this case, as shown in Figure 2, the algorithm is recursive.
- **Concept Covering**: many ubiquitous scenarios require that relatively large number of low-complexity resources are aggregated in order to satisfy an articulated request. To this aim, a further non-standard reasoning task based on the solution of *Concept Covering Problem* (CCoP, formally defined in (Ragone et al., 2007)) has been defined. It allows to: (i) cover (*i.e.*, satisfy) features expressed in a request as much as possible, through the conjunction of one or more instances of a KB –seen as elementary building blocks– and (ii) provide explanation of the uncovered part of the request itself. Given a concept expression D (request) and a set of instances $S = \{S_1, S_2, \ldots, S_n\}$ (available resources), where D and $S_1, S_2, \ldots, S_n$ are satisfiable in the reference ontology $\mathcal{T}$, *Concept Covering* aims to find a pair $\langle S_c, H \rangle$ where S_c includes concepts in S covering D w.r.t. $\mathcal{T}$ as much as possible and H is the residual part of D not covered by concepts in S_c.

Since S is an approximated match of D, it would be useful to evaluate how good the approximation is. Based on the uniqueness of CNF, a *norm* for concept expressions can be computed by "counting" the number of conjuncts in it (Ruta et al., 2011). Hence, numerical *penalty functions* can be defined based on the norm of expressions G and H_K, which allow to evaluate the goodness of match approximation as well as to rank several resources w.r.t. a request. Algorithms in Figure 1 and 2 compute and return associated penalty values.

In order to use Mini-ME in more general knowledge-based applications, the following reasoning services over on-

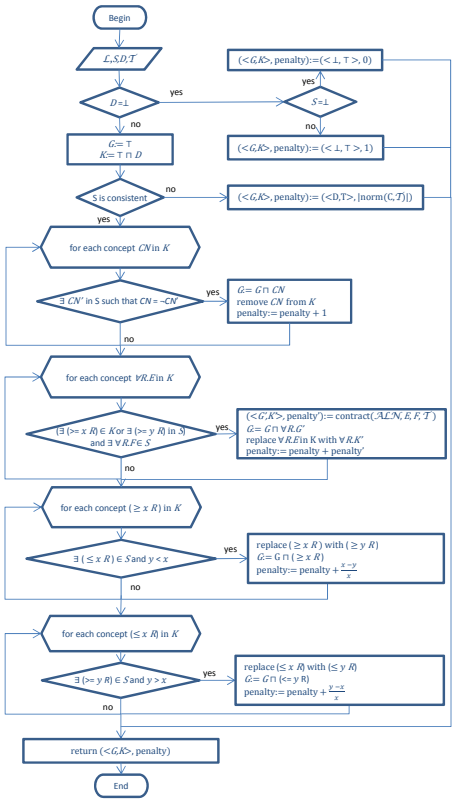


Figure 1. Concept Contraction algorithm

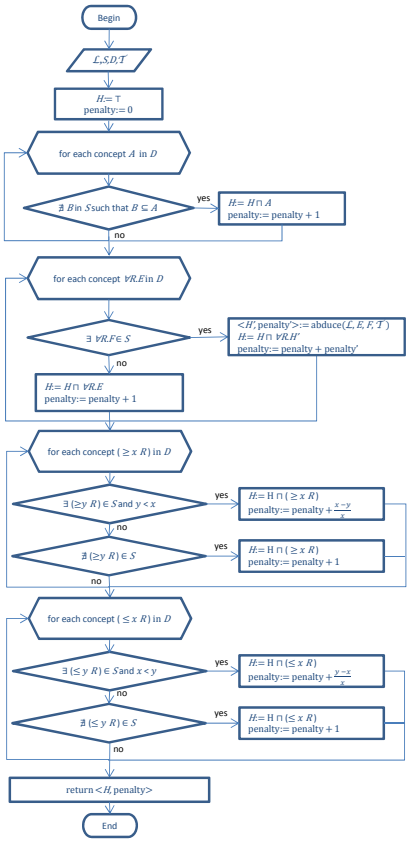


Figure 2. Concept Abduction algorithm

tologies were also implemented:
- **Ontology Coherence**: it is a simplified check w.r.t. Satisfiability, because it does not process ABox individuals (the difference is discussed *e.g.*, in (Moguillansky, Wassermann, & Falappa, 2010)). Since CNF normalization allows to identify unsatisfiable concepts, it is sufficient to normalize every concept during ontology parsing to locate unsatisfiabilities in the ontology.
- **Classification**: ontology classification computes the overall concept taxonomy induced by the subsumption relation, from $\top$ to $\bot$ concept. In order to reduce the subsumption tests, the following optimizations introduced in (Baader et al., 1994) were implemented: *enhanced traversal top search*, *enhanced traversal bottom search*, exploitation of *told subsumers*.

Architecture

Mini-ME architecture is sketched as UML diagram in Figure 3. Components are outlined hereafter:
- **Android Service**: implements a service (*i.e.*, a background daemon) any Android application can invoke to use the engine;
- **OWL API** (Horridge & Bechhofer, 2009): provides support for parsing and manipulating the OWL 2 language expressions;
- **OWLlink API** (Noppens, Luther, & Liebig, 2010): provides support for the core OWLlink (Liebig et al., 2011) protocol, thus allowing requests for standard inferences only;
- **MicroReasoner**: reasoner implementation, exposing fundamental KB operations (load, parse), as well as inference tasks;
- **KB Wrapper**: implements KB management functions (creation of internal data structures, normalization, unfolding) and inference procedures on ontologies (classification and coherence check);
- **Data Structures**: in-memory data structures for concept manipulation and reasoning; at this level inference procedures on concept expressions (concept satisfiability, subsumption, abduction, contraction, covering) are implemented.

Mini-ME was developed in Java using Android SDK Tools[6], Revision 12, corresponding to Android Platform version 2.1 (API level 7), therefore it is compatible with all devices running Android 2.1 or later. Mini-ME can be used: (a) through the *Android Service* by Android applications; (b)

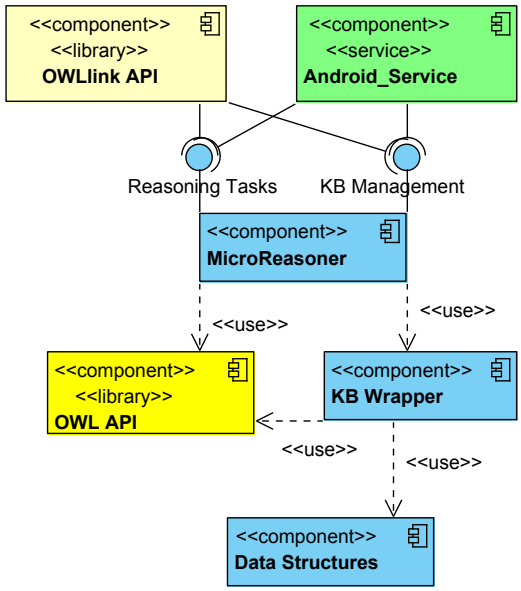


Figure 3. UML component diagram

via OWLlink; (c) as a library by calling public methods of the *MicroReasoner* component directly. In cases (b) and (c), it runs unmodified on Java Standard Edition runtime environment, version 6 or later. The system supports all OWL 2 syntaxes accepted by the OWL API parser.

Standard Java Collection Framework objects are used to define the low-level *Data Structures* package (mentioned before), composed of the following classes:

- **Item**: it represents a named concept expression. When parsing an ontology, the *KB Wrapper* component builds a Java *HashMap* object containing all concepts in the TBox as *String-Item* pairs. Each concept is unfolded, normalized and stored in the HashMap with its name as key and *Item* instance as value.
- **SemanticDescription**: models a concept expression in CNF as aggregation of $C_{CN}, C_{GT}, C_{LT}, C_{\forall}$ components, each one stored in a different Java *ArrayList*. Methods implement inference services.
- **Concept**: superclass of all concept types. Subclasses are **AtomicConcept**, **UniversalRestriction** and **CardinalityRestriction**, which is further extended by **GreatherThanRestriction** and **LessThanRestriction**. The `equals` method, inherited from *java.lang.Object*, has been overridden in order to properly implement logic-based comparison.
- **Abduction** and **Contraction**: represent the result returned by Concept Abduction and Concept Contraction, respectively. *Abduction* contains a *SemanticDescription* as *Hypothesis*, while *Contraction* contains two *SemanticDescriptions* as *Give Up* and *Keep*. Furthermore, they both contain a penalty score induced by the inference procedure.
- **Composition**: represents the result returned by the Concept Covering service. It contains a vector of *Items* as covering set and a further one as the uncovered part of the request.

Usage scenarios

Mini-ME can be used in both semantic and ubiquitous semantic web scenarios. The tool assists users in knowledge bases engineering, by providing consistency check and classification tasks. Furthermore, it provides resource discovery capabilities –through standard and non-standard inference services– to build applications and services for e-commerce, e-learning, knowledge management, healthcare and life sciences, among other applications. The possibility to run the proposed system on mobile and embedded devices allows to leverage semantics also in ubiquitous and pervasive contexts such as m-commerce, m-learning, ubiquitous health monitoring and healthcare, home and building automation, navigation and driving assistance systems, VANETs (Vehicular Ad-hoc NETworks), WSSANs (Wireless Semantic Sensor and Actor Networks), and many more. To focus just on few clear examples, this section illustrates how Mini-ME can be used for knowledge engineering with an ontology editor and in a typical ubiquitous computing application, a mobile augmented reality explorer to discover points of interest.

In the Semantic Web: Protégé plugins

Mini-ME has been integrated within the Protégé ontology editor through the implementation of an OWL reasoner plugin, in order to facilitate ontology engineering and allow to design and prototype ubiquitous knowledge-based applications. Standard reasoning tasks are accessible through the Protégé user interface in the *Reasoning* menu. The entry point is the *MinimeReasoner* class that extends *AbstractProtegeOWLReasonerInfo*, which manages the synchronization between the KB and the reasoner in case of changes to the currently loaded ontology. The *MinimeReasoner* class also references the reasoner factory, responsible for the creation of the *MicroReasoner* Mini-ME instance. Furthermore, the implemented OWLlink API (Noppens et al., 2010) server adapter allows to test the Mini-ME reasoner as an HTTP/XML OWLlink server, but only the core protocol is supported, *i.e.*, non-standard inference services are currently unsupported via OWLlink. By means of the OWLlink Protégé plugin configuration panel the user can set up the URL where the server is listening for incoming requests.

A further Protégé plugin has been developed to: (i) exploit non-standard inferences through a user-friendly GUI; (ii) support users during the development of ontology for pervasive and ubiquitous scenarios. The existing *DL Query*[7] plugin was used as guideline. The proposed plugin is a *Tab Widget* and it consists of the following components, highlighted in Figure 4: (A) *OWLIndividualsList* and *OWLIndividualsTypes* tabs, showing all KB instances with related description; (B) *OWLAssertedClassHierarchy* and *OWLClass-*

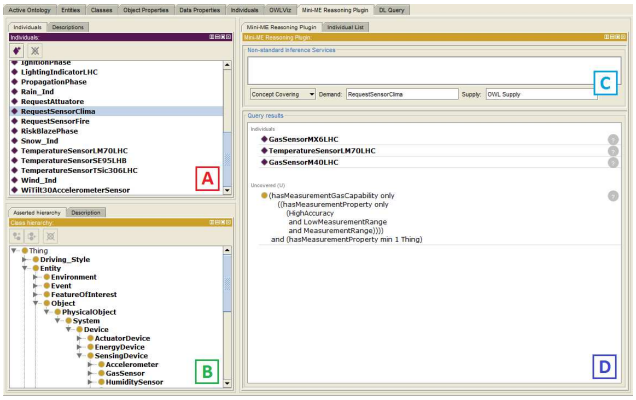


Figure 4. Protégé plugin for non-standard inferences

Description tabs, containing the general taxonomy along with the description of selected classes; (C) an input box used to select the inference task, the request *R* and –in case of abduction and contraction– the resource annotation *S* from the *OWLIndividualsList* through drag-and-drop. For Concept Covering, a subset of KB individuals can be checked through the *Individuals List* panel as composing resources. Finally, the results area (D) shows the output of the chosen inference service. In Figure 4 a CCoP is solved, component individuals and the uncovered part of the request are shown.

In the Ubiquitous Semantic Web: mobile augmented reality explorer

Semantic-based technologies can support articulated and meaningful descriptions of locations and Points of Interest (POIs). The use of metadata (annotations) endowed with formal machine-understandable meaning can enable advanced location-based resource discovery through proper inferences. The Mini-ME engine powers a novel discovery tool in Mobile Augmented Reality (MAR) for Android (Ruta, Scioscia, De Filippis, et al., 2014). The overall architecture of the proposed ubiquitous POI discovery framework is depicted in Figure 5. It consists of the following components:

- The *OpenStreetMap server* working as cartography provider. OSM map entities are semantically enriched in a way that best fits location-based resource discovery.
- A general method and an *editor* (Scioscia, Binetti, Ruta, Ieva, & Di Sciascio, 2014) for annotating maps, so allowing a collaborative crowd-sourced enrichment of basic OpenStreetMap cartography. The standard OWL 2 languages are exploited to create and share POI annotations, based on ontologies providing the conceptual vocabulary to express them.
- A MAR client providing the following features: (i) discovery of most relevant POIs w.r.t. user's semantically annotated profile, via a logic-based matchmaking; (ii) visualization of POI annotations and examination of discovery results, through a fully visual user interface.

In order to allow users to store semantic annotations in a POI description retaining backward compatibility, new tags have been introduced complying with the basic key-value pair stricture of OSM element tags:
`<tag k="semantic:n:key" v="value" />`
The semantic prefix is used to distinguish semantic annotations from other tags. The index `n` identifies different annotations –possibly referring to different ontologies– associated to the same map element. Key name suffix and value format differ for each proposed tag type, as in what follows:
`<tag k="semantic:n:ontology" v="URI" />` denotes the ontology the semantic node annotation refers to. The tag value is the unique ontology URI (Uniform Resource Identifier), which usually consists of a URL (Uniform Resource Locator) which can be accessed to retrieve the ontology.
`<tag k="semantic:n:encoding" v="format" />` specifies the compression format used to encode the semantic annotation. Compression techniques are needed in order to cope with the well-known verbosity of XML-based ontological languages such as RDF and OWL (Scioscia & Ruta, 2009).
`<tag k="semantic:n:counter" v="data" />` tags contain the Base64 string representation of the compressed semantic annotation. If its length is within 255 characters, a single tag is used, else it is split in 255-character segments and each one is stored in a tag. The counter suffix is assigned as a segment index, starting from 1.

The client –illustrated in Figure 6a– was developed using Android SDK Tools, Revision 23, corresponding to Android Platform version 4.2.2 (API level 17). The system adopted a modified version of the Android Augmented Reality framework[8]. Given POI target coordinates (latitude, longitude and altitude), it collects the azimuth and inclination angle between the device and the target from gyroscope and compass, in order to calculate where the device is pointing and its degree of tilt. Using this information, the system decides if and where a POI marker should be displayed within the viewfinder image on the screen.

In the proposed AR POI discovery framework, the user profile plays the role of request *R*. It consists of a concept expression including personal information like interests and hobbies. The profile is either composed by browsing visually the ontology modeling the reference domain (Scioscia et al., 2014), or imported from other applications and services. Available resources are the annotated OSM POIs in the user's area, referring to the same ontology as the user profile. They are extracted from OSM server and cached in the MAR client. Several resource domains (cultural heritage, shopping, accommodation, etc.) can be explored by simply selecting the proper reference ontology. Hence the

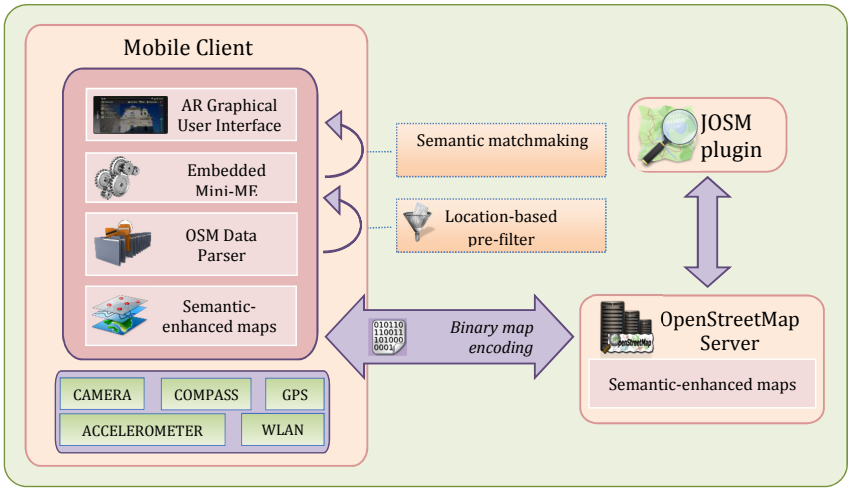
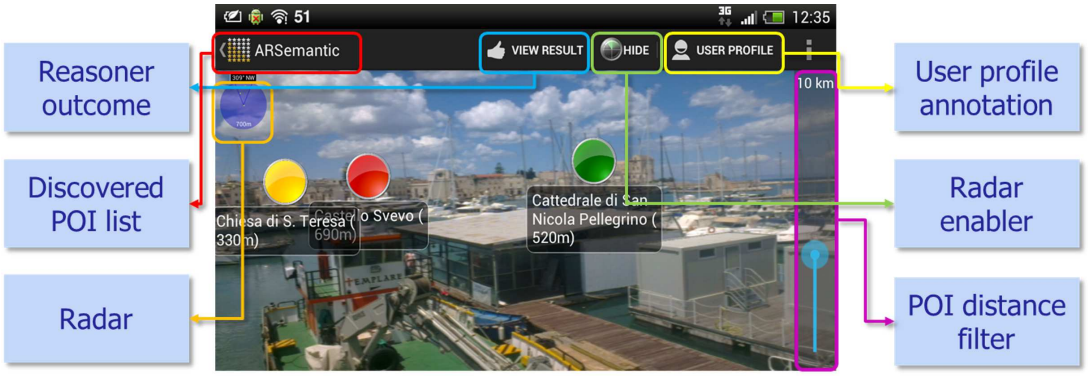


Figure 5. Architecture of the MAR System



(a) User interface

(b) Abduction results

(c) Contraction results

Figure 6. User interface of the semantic MAR explorer

proposed system works as a general-purpose location-based service discovery facilitator. Exploiting the embedded Mini-ME matchmaker, the mobile tool executes semantic matchmaking between the user profile and the annotations of POIs –enclosed into semantic-enhanced OSM map– in her surroundings, in a reference range with respect to user's position. Figure 7 sketches the resource discovery process. The semantic description concerning each POI is stored as an attribute of its marker. A score is finally given to each POI, expressing the result of the matchmaking between the user profile and the POI itself. The overall resource score is computed with the utility function:

$$f(R, POI) = 100[1 - \frac{penalty(R,POI)}{penalty(R,\top)}(1 + \frac{distance(User\_GPS, POI\_GPS)}{max\_distance})]$$

where $penalty(R, POI)$ is the semantic distance between profile R and POI; this value is normalized dividing by $penalty(R, \top)$, which is the distance between R and the universal concept and depends only on assertions in the ontology. Geographical distance (normalized by user-specified maximum range) is combined as weighting factor. The purposes of the utility function are to weight the result of semantic matching according to distance and to convert the score to a more user-friendly scale. Of course nearer resources are preferred, but in case of a full match $penalty(R, POI) = 0$ hence $f(R, POI) = 100$ regardless of distance.
By touching a marker, the user can see its relevant features, which are presented as icons around a wheel shape, in order to provide a clear and concise description, as shown in the central portion (A) of Figure 6b. The View result panel (B) in Figure 6b lists all missing features w.r.t. user profile (C), computed through Concept Abduction. In case of incompatibility, the same left-hand menu shows Concept Contraction outcome: properties the POI satisfies and incompatible elements (Figure 6c-(D)). Overall, the user can quickly identify what POI resources are most relevant to her needs and desires, by looking at the POI marker color, at the matchmaking result shown in the score panel and -if interested- by exploring POIs features. Simple operations on the device touchscreen allow effortless information acquisition and management.

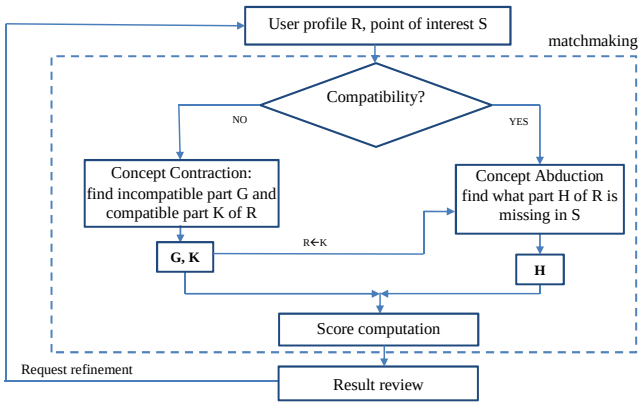


Figure 7. POI matchmaking process in the MAR system

Performance evaluation

An experimental campaign was performed both on PC and mobile devices, for standard and non-standard inference services. Methods and results are outlined hereafter. Full results about performance evaluation are reported on the Mini-ME home page[1].

Standard inference tests on PC

Mini-ME was compared on PC with FaCT++[9], HermiT[10] and Pellet[11]. All reasoners were used via the OWL API on a PC testbed (Intel Core i7 CPU 860 at 2.80 GHz –4 cores/8 threads– with 8 GB DDR3-SDRAM (1333 MHz) memory, 1 TB SATA (7200 RPM) hard disk, 64-bit Microsoft Windows 7 Professional and 32-bit Java 7 SE Runtime Environment, build 1.7.0_03-b05). The reference dataset was taken from the 2012 OWL Reasoner Evaluation workshop[12]: it is composed of 214 OWL ontologies with different size, expressiveness and syntax. For each reasoning task, two tests were performed: (i) correctness of results and turnaround time; (ii) memory usage peak. For turnaround time, each test was repeated four times and the average of the last three runs was taken. For memory tests, the final result was the average of three runs.

Classification. The input of the *classification* task was the whole ontology dataset. For each test, one of the following possible outcomes was recorded:

- *Correct*, the computed taxonomy corresponds with the reference classification (if it is available into the dataset) or results of all the reasoners are the same. In this case the total time taken to load and classify the ontology is also recorded;
- *Parsing Error*, the ontology cannot be parsed by the OWL API due to syntax errors;
- *Failure*, the classification task fails because the ontology contains unsupported logic language constructors;
- *Out of Memory*, the reasoner generates an exception due to memory constraints;
- *Timeout*, the task did not complete within the timeout threshold (set to 60 minutes).

Mini-ME correctly classified 83 of 214 ontologies; 71 were discarded due to parsing errors, 58 presented unsupported language constructors, the timeout was reached in 2 cases. Pellet classified correctly 130 ontologies, HermiT 128, FaCT++ 122. The lower "score" of Mini-ME is due to the presence of General Concept Inclusions, cyclic TBoxes or unsupported logic constructors. Parsing errors occurred in the OWL API library and were therefore common to all reasoners.

Performance was measured only for the 73 ontologies correctly classified by all reasoners, dividing them in five categories, based on the number of concepts:

- Extra Small (XS): fewer than 10 concepts; 13 ontologies were in this group;
- Small (S): between 10 and 100 concepts; 9 ontologies;
- Medium (M): between 100 and 1000 concepts; 25 ontologies;
- Large (L): between 1000 and 10000 concepts; 22 ontologies;
- Extra Large (XL): more than 10000 concepts; 4 ontologies.

Figure 8 compares the classification times of each reference

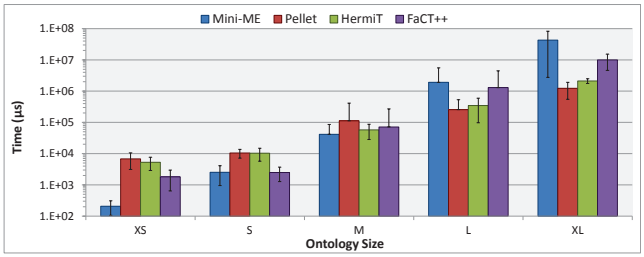


Figure 8. Classification test on PC

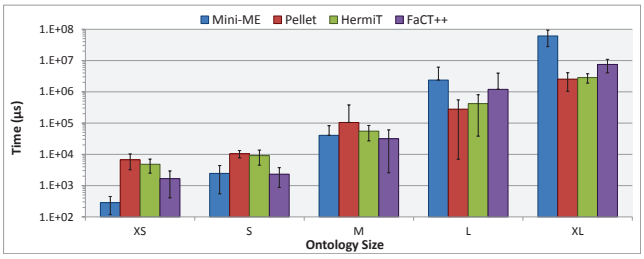


Figure 10. Ontology Satisfiability on PC

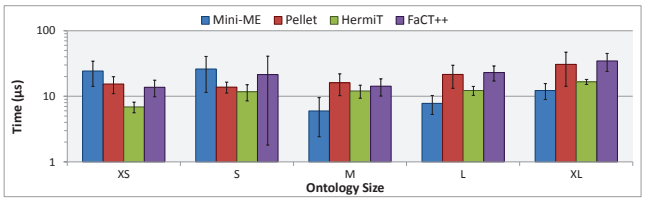


Figure 9. Class Satisfiability on PC

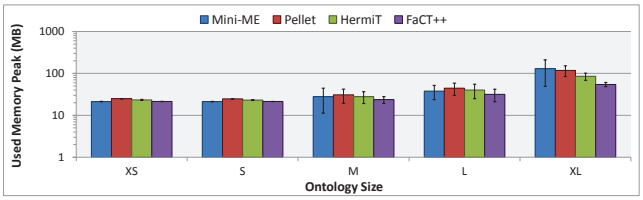


Figure 11. Memory usage test on PC

reasoner w.r.t. the ontology categories. Pellet, HermiT and FaCT++ exhibited a similar trend, with the first two faster than the other engines for large ontologies. Mini-ME is very competitive for small-medium ontologies (up to about 1200 classes) but less for large ones. This can be considered as an effect of the Mini-ME design, which is optimized to manage elementary TBoxes.

For *class satisfiability*, the dataset consists of 107 ontologies but only the 69 ontologies that Mini-ME correctly classified in the previous test were considered; for each of them, one or more classes were checked, as specified in the dataset. As reported in Figure 9, performances are basically similar, with times differing only for few microseconds and no reasoner consistently faster or slower. Moreover, the chart suggests no correlation between the time and the ontology size, whereas it is in direct ratio to the complexity of the class description and to its depth in the taxonomy (data not shown).

Figure 10 shows results for *ontology satisfiability*. For this task, all reasoners presented performance similar to the ones reported in Figure 8 for classification. In fact ontology satisfiability test implies loading, classifying and checking consistency of all concepts in the ontology with the first two steps requiring the larger part of the time. Outcomes of all reasoners are the same, except for ontologies with IDs 199, 200, 202, 203. In contrast to Pellet, HermiT and FaCT++, Mini-ME checks ontology coherence regardless of the ABox. The above ontologies include an unsatisfiable class (`GO_0075043`) with no instances, therefore the ontology is reported as incoherent by Mini-ME but as satisfiable by the other reasoners.

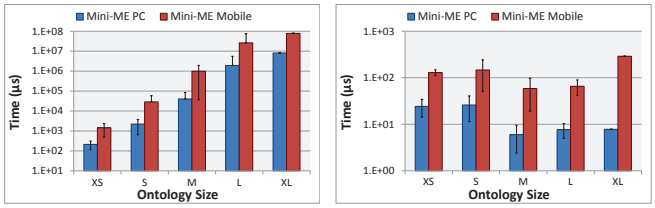
Finally, Figure 11 reports on *memory usage* peak during classification, which was verified as the most memory-intensive task. For small-medium ontologies, used memory is roughly similar for all reasoners. Mini-ME provides good results, with slightly lower memory usage than Pellet and HermiT and on par with FaCT++. Also for large ontologies Mini-ME results are comparable with the other reasoners, but in this case FaCT++ has the best performance.

Standard inference tests on mobile

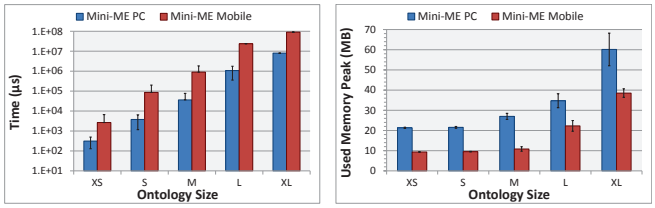
Using as reference the same ontology dataset, results obtained on an Android smartphone (Samsung Galaxy Nexus GT-I9250 with dual-core ARM Cortex A9 CPU at 1,2 GHz, 1 GB RAM, 16 GB storage memory, and Android version 4.2.2) were compared to the above outcomes for PC tests.

Results computed by Mini-ME on the Android smartphone were in all cases the same as on the PC. On the mobile device 75 ontologies out of 214 were correctly classified, 55 were discarded due to parsing errors, 56 had unsupported language constructors, 26 generated out-of-memory exceptions; (these included ontologies correctly classified on PC or not classified due to parsing errors or unsupported constructors) and 2 reached the timeout. Figure 12a shows a comparison between the classification turnaround times on PC and on mobile. Data refer to the 73 ontologies correctly classified by Mini-ME on both platforms.

Times are roughly an order of magnitude higher on the Android device. Absolute values for small-medium ontologies are under 1 second, so they can be deemed as acceptable in ubiquitous contexts. Furthermore, it can be noticed that the turnaround time increases linearly w.r.t. number of classes both on PC and on smartphone, thus confirming that Mini-ME has predictable behavior regardless of the reference platform. Similar considerations apply to class and ontology satisfiability tests: the turnaround time comparisons are re-



(a) Classification (b) Class Satisfiability



(c) Ontology Satisfiability (d) Memory Usage

Figure 12. Memory usage test on Mobile

ported in Figure 12b and Figure 12c, respectively.

Moreover, memory allocation peak during the classification task are reported Figure 12d. Data were obtained exploiting the *Android logging system*, which provides a mechanism for collecting and viewing system debug output, including heap memory data and garbage collector calls. For small-medium ontologies, the required memory was roughly fixed. Instead, for large ontologies the used memory increased according to the total number of classes. Moreover, in every test memory usage on Android was significantly lower than on PC. This is due to the harder memory constraints on smartphones, imposing to have as much free memory as possible at any time. Consequently, Android Dalvik virtual machine performs more frequent and aggressive garbage collection w.r.t. Java SE virtual machine. This reduces memory usage, but on the other hand can be responsible for a significant portion of the PC-smartphone turnaround time gap that was found.

Finally, a comparison was carried out using the five ontologies tested in (Yus et al., 2013). Mini-ME did not classify *Wine* and *Pizza* ontologies correctly because of unsupported language constructors, *DBpedia* produced a runtime error due to the presence of General Concept Inclusions, whereas *GO* and *NCI* gave an out-of-memory exception even using the *android:largeHeap="true"* attribute in the application manifest (other reasoners produced the same error as described in the referenced paper).

Non-standard inference tests

Performance evaluation was carried out for non-standard inferences using the same PC testbed and Android smartphone adopted for the previous tests. In this case the goal was to compare the performance trends on the two different

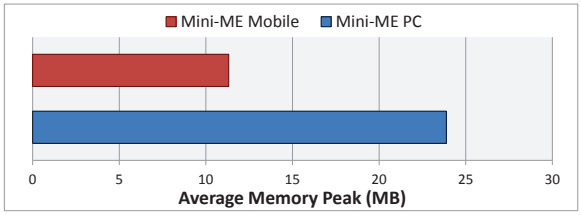


Figure 14. Memory usage for non-standard inferences

platforms, regardless of the specific PC or smartphone configuration. The test performed both unfolding and normalization over a 557 kB knowledge base with 100 request/resource pairs –randomly generated starting from the ontology defined in (Ruta et al., 2011), with average size of 4.2 kB– and finally executed abduction and contraction between each pair. Every task was repeated four times and the average turnaround time of the last three runs was taken. Figure 13 reports on time results (in microseconds) in case of PC and mobile testbed, respectively.

For each request/resource the system checks for compatibility; in case, abduction is performed, otherwise contraction is run, followed by abduction with the compatible part of the request. Notice that the computational time basically varies depending on the complexity of the semantic descriptions. Results for mobile tests have been referred to the ones for PC in order to highlight non-standard inferences exhibit similar trends. Processing times are reported in a logarithmic scale: in spite of the performance gap between fixed and handheld architectures, reasoning tasks maintain an acceptable computational load also on mobile platforms. Times were roughly an order of magnitude higher on the Android device. This is due not only to the limited computational capabilities of mobile devices, but also –as said above– to the more frequent garbage collection by the Android Dalvik virtual machine. On both platforms, all request/resource pairs show slightly variable memory peak values due to the similar structure of their semantic descriptions. For this reason Figure 14 reports only the average memory usage. Non-standard inferences on mobile require on average about 11.32 MB of memory with a standard deviation of 27 KB whereas on PC the average is about 23.88 MB with a standard deviation of 4 KB.

Conclusion

The paper presented a prototypical reasoner devised for ubiquitous computing. It supports Semantic Web technologies through the OWL API and implements both standard and non-standard reasoning tasks. Developed in Java, it targets the Android platform but also runs on Java SE and on embedded devices like Raspberry Pi. Experiments were performed both on PCs and smartphones and evidenced: (i) correctness of implementation, (ii) competitiveness with state-of-the-art reasoners in standard inferences, (iii) acceptable

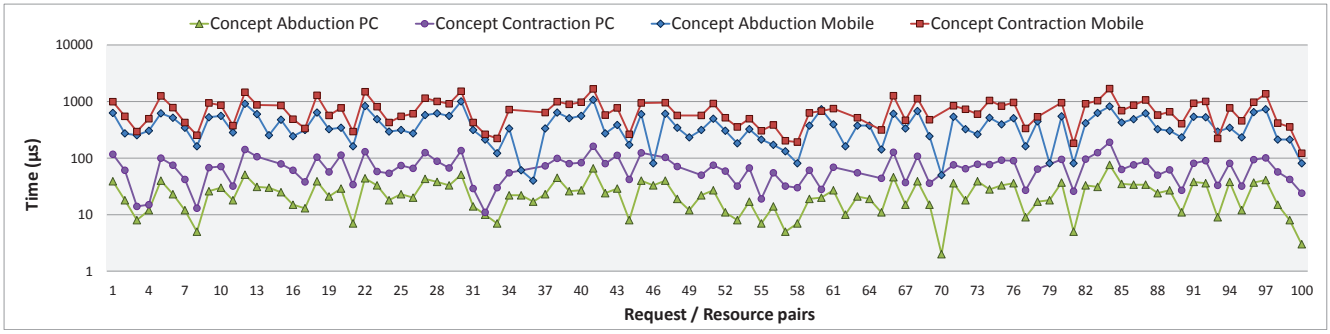


Figure 13. Non-standard inference tests

performance on target mobile devices. Besides further performance optimization leveraging peculiarities of Android Dalvik (for Android versions up to 4.4) and ART (for versions 5.0 and above) runtimes, future work includes: support for ABox management; the extension of OWLlink interface to non-standard inference services; implementation of further reasoning tasks; the support for more expressive languages, *e.g.*, with $\mathcal{EL}^{++}$extension of abduction and contraction algorithms.

The prototypical version of the software has been fruitfully used and tested in automotive scenarios and ambient intelligence ones. Some applications can be examined at http://sisinflab.poliba.it/swottools/. Therefore a widespread exploitation of it is foreseen in a project found under the Apulia Region Cluster research program in the healthcare field. Also a couple of Italian big companies have shown their interest for the early version of the engine for an extensive application of it in their products or services. At the moment we plan to release the reasoner under the Creative Commons license, but this choice will depend on its final usage.

Acknowledgement

The work was supported by Italian PON projects Puglia@Service and Puglia Digitale 2.0 and by E.T.C.P. Greece-Italy ARGES (pAssengeRs and loGistics information Exchange System) project.

References

Avancha, S., Joshi, A., & Finin, T. (2002, June). Enhanced Service Discovery in Bluetooth. *IEEE Computer*, *35*(6), 96–99.

Baader, F., Brandt, S., & Lutz, C. (2005). Pushing the EL envelope. In *International joint conference on artificial intelligence* (Vol. 19, p. 364).

Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., & Patel-Schneider, P. (2002). *The Description Logic Handbook*. Cambridge University Press.

Baader, F., Hollunder, B., Nebel, B., Profitlich, H., & Franconi, E. (1994). An empirical analysis of optimization techniques for terminological representation systems. *Applied Intelligence*, *4*(2), 109–132.

Baader, F., Lutz, C., & Suntisrivaraporn, B. (2006). CEL – a polynomial-time reasoner for life science ontologies. *Automated Reasoning*, 287–291.

Colucci, S., Di Noia, T., Pinto, A., Ragone, A., Ruta, M., & Tinelli, E. (2007). A Non-Monotonic Approach to Semantic Matchmaking and Request Refinement in E-Marketplaces. *Int. Jour. of Electronic Commerce*, *12*(2), 127–154.

Di Noia, T., Di Sciascio, E., & Donini, F. (2007). Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Jour. of Artificial Intelligence Research (JAIR)*, *29*, 269–307.

Di Noia, T., Di Sciascio, E., Donini, F. M., Ruta, M., Scioscia, F., & Tinelli, E. (2008). Semantic-based bluetooth-rfid interaction for advanced resource discovery in pervasive contexts. *Int. Jour. on Semantic Web and Information Systems (IJSWIS)*, *4*(1), 50–74.

Gennari, J. H., Musen, M. A., Fergerson, R. W., Grosso, W. E., Crubézy, M., Eriksson, H., ... Tu, S. W. (2003). The evolution of Protégé: an environment for knowledge-based systems development. *International Journal of Human-computer studies*, *58*(1), 89–123.

Haarslev, V., & Müller, R. (2001). Racer system description. *Automated Reasoning*, 701–705.

Horridge, M., & Bechhofer, S. (2009). The OWL API: a Java API for working with OWL 2 ontologies. *Proc. of OWL Experiences and Directions*, *2009*.

Horrocks, I., & Patel-Schneider, P. (1999). Optimizing description logic subsumption. *Jour. of Logic and Computation*, *9*(3), 267–293.

Kazakov, Y. (2009). Consequence-driven reasoning for Horn SHIQ ontologies. In *Ijcai* (Vol. 9, pp. 2040–2045).

Kazakov, Y., & Klinov, P. (2013). Experimenting with ELK Reasoner on Android. In *Proc. of 2nd international workshop on owl reasoner evaluation (oreâĂŹ13), ulm (germany)* (pp. 68–74).

Kazakov, Y., Krötzsch, M., & Simančík, F. (2014). The Incredible ELK. *Journal of Automated Reasoning*, *53*(1), 1–61.

Kim, T., Park, I., Hyun, S., & Lee, D. (2010). MiRE4OWL: Mobile Rule Engine for OWL. In *Computer software and applications conf. workshops (compsacw), 2010 ieee 34th annual* (pp. 317–322).

Kleemann, T., & Sinner, A. (2005). User Profiles and Matchmaking on Mobile Phones. In O. Bartenstein (Ed.), *Proc. of 16th int.*

conf. on applications of declarative programming and knowledge management inap2005, fukuoka.

Koch, F., Meyer, J.-J. C., Dignum, F., & Rahwan, I. (2006). Programming deliberative agents for mobile services: the 3APL-M platform. In *Programming multi-agent systems* (pp. 222–235). Springer.

Lawley, M., & Bousquet, C. (2010). Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In *Proc. of the 6th australasian ontology workshop. conferences in research and practice in information technology* (Vol. 122, pp. 45–49).

Levesque, H. (1984). Foundations of a Functional Approach to Knowledge Representation. *Artificial Intelligence*, *23*(2), 155–212.

Li, L., & Horrocks, I. (2004). A software framework for matchmaking based on semantic web technology. *International Journal of Electronic Commerce*, *8*(4), 39–60.

Liebig, T., Luther, M., Noppens, O., & Wessel, M. (2011). OWLlink. *Semantic Web*, *2*(1), 23–32.

Liu, Q. (2013). U–commerce research: a literature review and classification. *International Journal of Ad Hoc and Ubiquitous Computing*, *12*(3), 177–187.

Moguillansky, M., Wassermann, R., & Falappa, M. (2010). An argumentation machinery to reason over inconsistent ontologies. *Advances in Artificial Intelligence–IBERAMIA 2010*, 100–109.

Motik, B., Horrocks, I., & Kim, S. M. (2012). Delta-reasoner: a semantic web reasoner for an intelligent mobile platform. In *Proc. of the 21st international conference companion on world wide web* (pp. 63–72).

Noppens, O., Luther, M., & Liebig, T. (2010). The OWLlink API: Teaching OWL Components a Common Protocol. In *Owled* (Vol. 614).

Pfisterer, D., Romer, K., Bimschas, D., Hasemann, H., Hauswirth, M., Karnstedt, M., ... Truong, C. (2011). SPITFIRE: toward a Semantic Web of things. *Communications Magazine, IEEE*, *49*(11), 40–48.

Ragone, A., Di Noia, T., Di Sciascio, E., Donini, F. M., Colucci, S., & Colasuonno, F. (2007). Fully automated web services discovery and composition through concept covering and concept abduction. *International Journal of Web Services Research (JWSR)*, *4*(3), 85–112.

Ruta, M., Di Noia, T., Di Sciascio, E., Piscitelli, G., & Scioscia, F. (2007). RFID meets Bluetooth in a semantic based u-commerce environment. In *Proc. of the 9th international conference on electronic commerce* (pp. 107–116).

Ruta, M., Di Noia, T., Di Sciascio, E., Piscitelli, G., & Scioscia, F. (2008). A semantic-based mobile registry for dynamic RFID-based logistics support. In *Icec '08: Proc. of the 10th int. conf. on electronic commerce* (pp. 1–9). New York, USA: ACM. doi: http://doi.acm.org/10.1145/1409540.1409576

Ruta, M., Di Sciascio, E., & Scioscia, F. (2011). Concept abduction and contraction in semantic-based P2P environments. *Web Intelligence and Agent Systems*, *9*(3), 179–207.

Ruta, M., Scioscia, F., De Filippis, D., Ieva, S., Binetti, M., & Di Sciascio, E. (2014, jul). A semantic-enhanced augmented reality tool for OpenStreetMap POI discovery. In *17th meeting of the euro working group on transportation (ewgt 2014).* (to appear)

Ruta, M., Scioscia, F., Di Noia, T., & Di Sciascio, E. (2009). Reasoning in Pervasive Environments: an Implementation of Concept Abduction with Mobile OODBMS. In *2009 IEEE/WIC/ACM Int. Conf. on Web Intelligence* (pp. 145–148). IEEE.

Ruta, M., Scioscia, F., Di Noia, T., & Di Sciascio, E. (2010, may). A hybrid ZigBee/Bluetooth approach to mobile semantic grids. *Computer Systems Science and Engineering*, *25*(3), 235–249.

Ruta, M., Scioscia, F., & Di Sciascio, E. (2010). Mobile Semantic-based Matchmaking: a fuzzy DL approach. *The Semantic Web: Research and Applications*, 16–30.

Ruta, M., Scioscia, F., Loseto, G., & Di Sciascio, E. (2014, feb). Semantic-based resource discovery and orchestration in Home and Building Automation: a multi-agent approach. *IEEE Transactions on Industrial Informatics*, *10*(1), 730–741.

Scioscia, F., Binetti, M., Ruta, M., Ieva, S., & Di Sciascio, E. (2014, feb). A Framework and a Tool for Semantic Annotation of POIs in OpenStreetMap. In *16th meeting of the euro working group on transportation (ewgt 2013)* (Vol. 111, pp. 1092–1101). Elsevier.

Scioscia, F., & Ruta, M. (2009). Building a Semantic Web of Things: issues and perspectives in information compression. In *Semantic web information management (swim'09). in proc. of the 3rd ieee int. conf. on semantic computing (icsc 2009)* (pp. 589–594). IEEE Computer Society.

Shearer, R., Motik, B., & Horrocks, I. (2008). HermiT: A highly-efficient OWL reasoner. In *Proc. of the 5th Int. Workshop on OWL: Experiences and Directions (OWLED 2008)* (pp. 26–27).

Sinner, A., & Kleemann, T. (2005, July). KRHyper - In Your Pocket. In *Proc. of 20th int. conf. on automated deduction (cade-20)* (p. 452-457). Tallinn, Estonia.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., & Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics: science, services and agents on the World Wide Web*, *5*(2), 51–53.

Steller, L., & Krishnaswamy, S. (2008). Pervasive Service Discovery: mTableaux Mobile Reasoning. In *Int. conf. on semantic systems (i-semantics). graz, austria.*

Tai, W., Keeney, J., & O'Sullivan, D. (2011). COROR: a composable rule-entailment owl reasoner for resource-constrained devices. *Rule-Based Reasoning, Programming, and Applications*, 212–226.

Tsarkov, D., & Horrocks, I. (2006). FaCT++ description logic reasoner: System description. *Automated Reasoning*, 292–297.

Vazquez, J. I., & López-de Ipiña, D. (2007). mRDP: An HTTP-based lightweight semantic discovery protocol. *Computer Networks*, *51*(16), 4529–4542.

von Hessling, A., Kleemann, T., & Sinner, A. (2004, September). Semantic User Profiles and their Applications in a Mobile Environment. In *Workshop on artificial intelligence in mobile systems (aims '04).*

Yus, R., Bobed, C., Esteban, G., Bobillo, F., & Mena, E. (2013). Android goes semantic: DL reasoners on smartphones. In *Proc. of 2nd international workshop on OWL reasoner evaluation (ore 13)* (pp. 46–52).

Footnotes

[1]http://sisinflab.poliba.it/swottools/minime

[2]OWL 2 Web Ontology Language, W3C Recommendation 11 December 2012, http://www.w3.org/TR/owl-overview/

[3]http://protege.stanford.edu/

[4]RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation 25 February 2014, http://www.w3.org/TR/rdf11-concepts/

[5]OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition), W3C Recommendation 11 December 2012)

[6]http://developer.android.com/sdk/tools-notes.html

[7]http://protegewiki.stanford.edu/wiki/DL_Query

[8]https://code.google.com/p/android-augment-reality-framework/

[9]Version 1.6.3 with OWL API 3.4, http://owl.man.ac.uk/factplusplus/

[10]Version 1.3.8, http://hermit-reasoner.com/

[11]Version 2.3.1, http://clarkparsia.com/pellet/

[12]http://www.cs.ox.ac.uk/isg/conferences/ORE2012/materials.html