Benchmarking results of semantic reasoners applied to an ambient assisted living environment ⋆

David Ausín, Federico Castanedo and Diego López-de-Ipiña

Deusto Institute of Technology, DeustoTech. University of Deusto.
Avda. Universidades 24. 48007 Bilbao, Spain.
{david.ausin, fcastanedo, dipina}@deusto.es
http://www.morelab.deusto.es/

Abstract. Ambient Assisted Living (AAL) environments and applications have been receiving a great amount of interest in recent years. The main reason of this interest is the increasing lifetime of elderly people. Our goal is to determine and clarify which combination of semantic reasoners and frameworks is the most suitable for building knowledge-driven smart environments. In particular, we are interested in the combination which provides a better computational performance and reasonable memory requirements. The obtained results show that the fastest choice under the employed dataset is to use the OWL API accessing the Pellet reasoner. On the other hand, the less memory consumption experiment is provided by a combination of HermiT and custom Java rules.

1 Introduction

A key component of an intelligent environment is how the user behavior is modeled and reasoned over time in order to assist the user in his activities of daily living. There are several ways to achieve this goal, ranging from statistics to logics and rule-based engines. Generally speaking, there are two broad different approaches for building an intelligent environment or activity recognition system. On the one hand, the *data driven* approach focuses on inferring the activities from the acquired data without using any a priori knowledge model, that is, without explicit knowledge. On the other hand, the *knowledge driven* approach relies on logic based methods. Well-known Description Logics (DL) provides high expressiveness and when combined with appropriate computational properties are the underlying theory for ontologies. The main drawbacks of this approach are: (i) it requires a priori knowledge engineering task about the world model representation; (ii) their difficulty in handling uncertainty due to the violation of some properties that produce undecidable logics [5].

In this work, we focus on the knowledge driven approach, therefore an AAL environment is modeled using ontologies and different semantic reasoners are applied to a simulated stereotypical scenario obtaining corresponding benchmarking results. The aim of this work is to determine and clarify which combination of reasoners and framework is the most suitable for building knowledge-driven smart environments. In

⋆ This work is supported by the Spanish MICINN project TALIS+ENGINE (TIN2010-20510-C04-03)

particular, we are interested in the combination which provides the best computational performance and reasonable memory requirements.

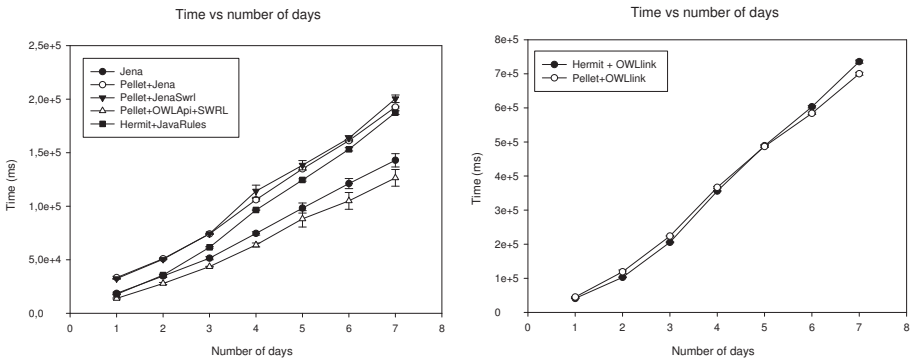

Fig. 1. Obtained results of reasoners execution time as we increase the number of days. Several reasoning tasks are performed every day. The left graph shows the results of the reasoners used as libraries whereas the one on the right side shows the results of OWLlink protocol access to reasoners (Client/Server approach).

2 Benchmark experiments and obtained results

An scenario using the seven days user activities dataset [3] has been employed. In this scenario, an elderly person, who needs help to remember when he must take his medicines, has been modeled.

A core and extension OWL 2 DL ontology have been developed and are the same for all the experiments. In addition to the ontologies, a set of rules has been created. These rules check if the user has taken the corresponding medicament, and advises him when he forgets to take it or when he takes the wrong one. These rules do not try to cover all the possible situations that can be produced, but do at least, the most important cases. Rules have been implemented using SWRL rules and Jena rules format. As rules execution are only available out-of-the-box in Pellet and Jena reasoners, custom Java rules have been developed for HermiT and reasoners used them via OWLlink protocol. These Java rules aim to provide the same inference results as SWRL and Jena rules. The experiments have been carried out using an Acer Aspire 4830TG laptop whose main features are an Intel Core i5-2410M and 8 GB of RAM. The operating system is an Ubuntu 11.10 for 64 bits, using Java 1.6.0_23 version provided by the OpenJDK Runtime Environment. The program firstly loads the ontologies into a reasoner. Then, it reads the simulated activities that the user is supposed to be doing from the dataset and updates the user context. Frameworks and reasoners combinations are summarized in the table 1. Each combination has been executed ten times and the data presented below shows the average and the standard deviation. Figure 1 shows the time needed by the reasoners whereas figure 2 the memory consumption.

	OWLlINK API 1.2.2 [4]	*OWL API 3.1.0* [1]	*Jena 2.6.4* [2]
Jena 2.6.4 [2]	-	-	(1) Jena rules
HermiT 1.3.5 [6]	(2) Java rules	(3) Java rules	-
Pellet 2.3.0 [7]	(4) Java rules	(5) SWRL rules	(6) Jena and (7) SWRL rules

Table 1. Different combinations of reasoners and semantic frameworks used in the benchmark experiments and the kind of rules employed.

As figure 1 shows the fastest choice is to use the OWL API to access to the Pellet reasoner. This result is pretty meaningful, because this approach is a 12% better than the second fastest configuration. However, the fastest configuration has also the biggest memory consumption, as figure 2 shows. In general, these figures do not show a relationship between the necessary time to perform a task and the memory consumption. Another important issue to remark is that in general the reasoner used in combination with the Jena framework (Pellet+Jena, Pellet+JenaSwrl in figure 1 left) is slower than others. However the time performance is improved when the reasoner provided by Jena is employed (Jena in figure 1). This issue could be the result of a more lightweight reasoning support for OWL in Jena reasoners.

We also measure the amount of used memory for the different combination. Memory used includes the memory occupied by all objects including both reachable and unreachable objects. The used memory results (see figure 2 left) show that the best memory consumption combination is to use HermiT+JavaRules through OWL API (case 3 of table 1). In the particular case of Pellet+OWLApi+SWRL (case 5 of table 1) we can notice a inverse relation between the memory consumption and the execution time, since it has the smallest execution time and the highest memory consumption.

In the case of using the OWLlink protocol to access reasoners capabilities, a high increment of execution time is noticed (see figure 1 right). This is because of the required marshalling/un-marshalling of the data and the added abstraction layer. The obtained results of memory consumption shows an equal amount of memory required in the client side (see figure 2 right) as we expected.

3 Conclusions and future work

In this work we have measured the performance of several semantic reasoners combined with semantic frameworks in a simulated AAL scenario. The results show that the performance of a reasoner can vary meaningfully depending on the semantic framework used. Regarding memory consumption, we have noticed that the combination of HermiT and Java Rules (see Figure 2 left) provides the best memory consumption. On the other hand, Pellet+OWLApi+SRWL provide the worst one. We suppose that this effect is due to the fact that Pellet implementation needs to transform the internal data from OWLApi format to the internal Pellet representation. In contrast, we suppose that HermiT provides better memory consumption results because it internally employs OWLApi format. An implementation using a distributed architecture might improve the time execution and could be something to be researched. One particular issue that

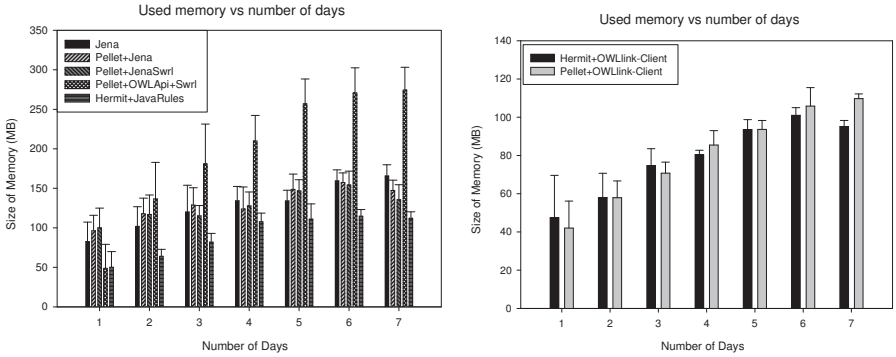

Fig. 2. Used memory results when we increase the number of days. Several reasoning tasks are performed every day. Memory used includes the memory occupied by all objects including both reachable and unreachable objects. The left graph shows the results of the reasoners libraries whereas the one on the right side shows the results of OWLlink protocol access to reasoners (Client side memory).

should be considered for AAL scenarios is that OWL 2 does not provide native support to handle uncertainty about the data being modeled. We believe there is a huge room to improve current semantic reasoners through reasoning support under uncertainty and there are few works covering this topic [5].

In the future, we would like to deepen into the particular implementation details of each reasoner/framework combination in order to have more information about the performance and memory consumption differences.

References

1. S. Bechhofer, R. Volz, and P. Lord. Cooking the semantic web with the owl api. *The Semantic Web-ISWC 2003*, pages 659–675, 2003.
2. J.J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson. Jena: implementing the semantic web recommendations. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 74–83. ACM, 2004.
3. T. Huynh, M. Fritz, and B. Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19. ACM, 2008.
4. Thorsten Liebig, Marko Luther, Olaf Noppens, and Michael Wessel. Owllink. *Semantic Web – Interoperability, Usability, Applicability*, 2(1):23–32, 2011.
5. T. Lukasiewicz and U. Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):291–308, 2008.
6. R. Shearer, B. Motik, and I. Horrocks. Hermit: A highly-efficient owl reasoner. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008)*, pages 26–27, 2008.
7. E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.