Optimised Mobile Reasoning for Pervasive Service Discovery

Luke Steller and Shonali Krishnaswamy
Centre for Distributed Systems and Software Engineering, Monash University, Melbourne, Australia
laste4@student.monash.edu.au,
Shonali.Krishnaswamy@infotech.monash.edu.au

Abstract

Today’s mobile users have access to a wide range of web-based services. This paper presents our m-Tableaux algorithm for enabling cost-efficient and optimised semantic reasoning to support pervasive service discovery. We present performance evaluation of the m-Tableaux optimisation strategies which clearly demonstrate its operational feasibility on a mobile device.

1. Introduction

Studies such as [1] have established that mobile users typically have a tolerance threshold of about 5 to 15 seconds in terms of response time, before their attention shifts elsewhere, depending on their environment. Thus, service discovery architectures that operate in mobile environments must cope with the very significant challenges of not merely finding relevant services, but being able to do so rapidly in a highly dynamic and varying context.

The limitations of syntactic, string-based matching for web service discovery coupled with the emergence of the semantic web implies that next generation web services will be matched based on semantically equivalent meaning, even when they are described differently and will include support for partial matching in the absence of an exact match. There is typically a need for a high-end computing resource for performing semantic service discovery. This reliance on a high-end, centralised node for performing for performing semantically driven pervasive service discovery can clearly be attributed to the fact that semantic reasoners used by these architectures are resource intensive. As such, they are unsuitable for deployment on small resource constrained devices, such as PDAs and mobile phones. These small devices which are typical in the context of mobile service discovery are quickly overwhelmed when the search space in terms of the size of ontologies and reasoning complexity increases. KRHyper [17] implements a First Order Logic (FOL) reasoner using Tableaux without the expected performance degradation that one would expect compared to DL (a decidable subset of FOL) reasoning. KRHyper performs better than Racer, however it is still quickly overwhelmed (as ontologies/complexity grows), out of memory exceptions occur and no response is provided. Clearly, this shows that such reasoners cannot be directly ported to a mobile device in their current form.

The reality of mobile environments is a world characterised by ad-hoc an intermittent connectivity where such reliance on remote/centralised processing (and continuous interaction) may not always be possible or desirable given the need for rapid processing and dynamically changing context (e.g. a driver has gone past a parking area). Pervasive service discovery has to necessarily be under-pinned by the current context to meet the all-important criteria of relevance in constantly changing situations. The communication overhead (not to mention the infeasibility/impracticability) of constantly relaying contextual and situational changes of the user/device to a central server will lead to inevitable delays. Thus, there is a clear imperative that for semantically driven pervasive service discovery to meet the very real response-time challenges of a mobile environment, the capacity to perform matching and reasoning must occur on the resource limited device itself. Therefore, there is a need for developing a pervasive services discovery architecture, which more flexibly manages the trade-off between computation time and precision of results, depending on the available resources on the device.

In this paper we present the implementation and evaluation of our mTableaux algorithm [theoretical details of algorithm available at Steller08a, Steller08b], which incorporates a weighted approach to reasoning and implements strategies to optimise DL reasoning tasks so that relatively large reasoning tasks of several hundred individuals and classes may function on small devices.

2. Implementation and Performance Evaluation

We implemented the two scenarios – a Printer Search scenario (Case study 1) comprising 141 classes, 337 individuals and 126 roles and a Movie Search Service (Case study 2) comprising 204 classes, 241 individuals and 93 roles.

In order to show how mTableaux compares to other commercial OWL semantic reasoners, we provide a performance comparison with RacerPro. Since RacerPro
is a desktop reasoner, which cannot be deployed to mobile devices, we have undertaken a performance evaluation on a Pentium Centrino 1.82GHz computer with 2GB memory with Java 1.5 (J2SE) allocated maximum of 500MB for each experiment. The results show that mTableaux performed in less than half the time of Pellet without classify for the Printer case study, with less substantial improvements for the Product ontology. It considerably outperformed RacerPro. These results are available in [Steller08b].

This paper presents our evaluation on a mobile device – a HP iPAQ hx2700 PDA, with Intel PXA270 624Mhz processor, 64MB RAM, running Windows Mobile 5.0 with Mysaifu Java J2SE Virtual Machine (JVM), allocated 15MB of memory. We executed four novel consistency checks, each 16 times, using various mTableaux optimisation strategies. Test 16 was the base case representing normal execution of the Tableaux algorithm, with none of our optimisations strategies enabled. This test resulted in the “Out Of Memory” exception illustrated in figure 1(a), in all cases except for individual A, in case study 2 where matching required over 2000 seconds.

![Figure 1. (a) Out of memory error (b) Correctly matched service URI returned.](image)

Figures 2 and 3 show two graphs, which each show the consistency time to perform a type check for individual A and B (indicated in the legend) against the request, and the optimisation overhead involved.

![Figure 2. Processing time taken to perform each test.](image)

![Figure 3. Optimisation overhead breakdown.](image)

**Conclusion and Future Work**

mTableaux was shown to significantly improve the performance of pervasive discovery reasoning tasks in two case studies, so that they can be completed on small resource constrained devices. It was also shown to out perform RacerPro without reducing the quality of results returned. We are also developing a resource-aware reasoning strategy to better manage the trade-off between result correctness and resource availability.

**References**

