Optimised Semantic Reasoning for Pervasive Service Discovery

Luke Steller and Shonali Krishnaswamy

Faculty of Information Technology, Monash University, Melbourne, Australia
laste4@student.monash.edu.au,
Shonali.Krishnaswamy@infotech.monash.edu

Abstract. A key challenge in delivering mobile services is to improve the relevance of discovered services, as mobile environments are very dynamic with rapid changes to user context. This paper presents m-Tableaux - an to optimised semantic reasoning approach to support pervasive service discovery which aims to efficiently leverage the computational resources available of mobile devices. We present performance evaluation of the m-Tableaux optimisation strategies which clearly demonstrate its operational feasibility on a mobile device.

1 Introduction and Related Work

A mobile user arriving in Sydney airport can currently utilise kiosk touch screens to search for stores and points of interest in the airport. The increasing computational capacity of small devices such as PDAs and mobile phones provide new opportunities for "on board" service discovery taking user context and request complexity into consideration, rather than limited and inconvenient fixed point kiosks. There are two models for discovery in this context: 1. the kisosk becomes a centralised high-end server which performs all matching on behalf of the user or 2. a decentralised model where the kiosk acts only as a repository of information and the user's device performs all matching "on-board", on a needs basis. We advocate the decentralised model for environments where a central authority does not exist or does not want the responsibility of providing and maintaining a centralised service and the user's do not want to incur the costs in using such a service.

While current service discovery architectures such as Jini [1] and UPnP [2] use either interface or string based syntactic matching, there is a growing emergence of DAML-S/OWL-S semantic matchmakers such as CMU Matchmaker [3] and DIANE [4] which support varying levels of semantic reasoning, but require a centralised high-end node to perform reasoning. The reasoners which matchmaking architectures use, such as FaCT++ [5] and RacerPro [6]), quickly give "Out of Memory" errors when ported directly to resource limited devices in their current form. In this paper, we present our mTableaux algorithm with incorporates optimisation strategies to enable reasoning on resource constrained mobile devices. Section 2 describes our architecture and our mTableaux algorithm, section 3 provides a performance evaluation and in section 4 we conclude the paper.

A. Bouguettaya, I. Krueger, and T. Margaria (Eds.): ICSOC 2008, LNCS 5364, pp. 620–625, 2008.

2 Resource-Aware and Cost-Efficient Pervasive Service Discovery

Our decentralised pervasive service discovery architecture is illustrated in figure 1, which resides on the user's resource constrained device. The remainder of this paper concentrates on the semantic mTableaux, while the adaptive discovery manager is future work.

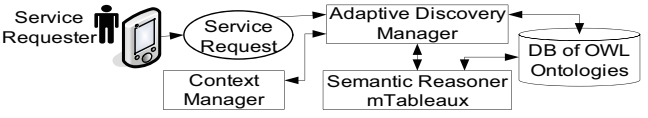

Fig. 1. Pervasive Service Discovery Architecture

2.1 Semantic Reasoners

The effective employment of semantic languages requires the use of semantic reasoners such as Pellet [7], FaCT++ [5] and RacerPro [6]. Most of these reasoners utilise the widely used Tableaux [8] algorithm. DL Tableaux reasoners, such as Pellet, reduce all reasoning tasks to a consistency check. Tableaux is a branching algorithm, in which disjunctions form combinations of branches in the tree. If all branches contain a clash, meaning a fact and its negation are both asserted, then a clash exists for all models of the knowledge base. Inferred membership for an individual I to class type RQ: I $\in$ RQ, is checked by asserting that I is a member of the negation of RQ. If a clash exists for all models of the knowledge base then the membership is proven.

This paper concentrates on strategies to optimise the Tableaux algorithm to enable reasoning on small/resource constrained devices with significant improvements to response time and avoiding "Out of Memory" errors as encountered in [9]. Our mTableaux algorithm relates to optimising membership type checks (I $\in$ RQ). It involves a range of optimisation strategies such as: 1. selective application of consistency rules, 2. skipping disjunctions, 3. establishing pathways of individuals and disjunctions which if applied would lead to potential clashes and associating weights values to these elements such that the most likely disjunctions are applied first, by 3a. ranking individuals and 3b. ranking disjunctions.

Application of consistency rules to a subset of individuals only, reduces the reasoning task. Consider the universal quantifier construct of the form $\forall$R.C = $\{$ b.(a, b) $\in$ R $\rightarrow$b C$\}$ [10], where R denotes a relation and C denotes a class concept, which implies object fillers for R must be of type C. We define the individual subset to I and those individuals which fill R for I. Disjunctions are only applied when they contain a concept which is contained within or can be unfolded from the request type RQ, including quantifier role filler types. A weighted queue is established to rank individuals and disjunctions such that the highest are applied first. Rankings are established by recursively checking

the class types contained in a disjunction to see whether these potentially lead to future clash. In the next section we more formally describe each of these strategies.

2.2 mTableaux Strategies

Selective Consistency: Let S denote a set of individuals e which can have completion rules applied to them, S ≡ {e_1, e_2, e_n}. S initially contains individual I. Let avR denote the distinct set of roles r contained in any universal quantifier construct which will be applied to any individual e in S, where avR ≡ {r_1 ,r_2 , r_m). Add any object individual o which fills any role r contained within avR, to S. The same strategy is recursively applied to all individuals added to S, or whenever a new universal quantifier is applied to an individual in S. For example consider that individual LaserPrinter1 has a role hasComm which contains the individual Modem1. Initially LaserPrinter1 is contained in S, and the assertion of ∀hasComm.¬(Modem), results in Modem1 also being added to S.

Disjunction Skipping: Where a service request RQ is a conjunction class type, let T denote the set of conjunct terms t, where T ≡ {t_1, t_2, t_n}. Let DS denote a set of types derived from RQ. For each t_i in T, add $_i$t to DS as well as all expressions which can be unfolded from t_i. Where any universal or existential quantifier, or cardinality restriction is encountered, add its role filler type C and all expressions which can be unfolded from C. During reasoning apply only those disjunctions D, containing disjunct terms d, where any d is contained within DS, otherwise skip it. For example, where RQ ≡ SupportsModem ⋂ SupportsColour, and SupportsModem ≡ ∃hasComm.Modem, DS ≡ {SupportsModem, SupportsColour, Modem}.

Weighted Individuals and Disjunctions: A single weighted queue QI of individuals i is established and each individual has a weighted queue QD of disjunctions d, such that QD ≡ {i_1, i_2, i_n} and QD ≡ {d_1, d_2, d_m} and there are n QDs. Each element i and d, is associated with a weight value which is used to rank the elements in decending order. Disjunctions with the highest weight in the QD, which relates to the individual with the highest weight in QI, are applied first. For any disjunct type element c of a disjunction D relating to individual I (rank disjunctions) or any type c applied to individual I (arising from the application of a disjunction) which did not give rise to a clash (rank individuals), search for a pathway to a potential clash. This occurs by checking to see if the application of any expression which can be unfolded from c, may give rise to a future clash. Where c unfolds into a disjunction or conjunction, the negation of their constitute elements is checked for in individual I. Where the expression is a universal quantifier, each object individual for I is checked for a potential clash for the quantifier's role filler. If potential clash path is detected the weight of all individuals and disjunctions involved in the path is increased. For example, for the disjunction. D ≡ SupportsModem ⋃ SupportsColour where SupportsModem ≡ ∀hasComm.Modem and Individual I has the type ¬Modem a potential clash is found and the weighting for disjunction D and individual I is increased in their queues, QD and QI respectively.

3 Implementation and Performance Evaluation

In this section we provide two case studies in order to evaluate our mTableaux algorithm. In case study 1, Bob is walking around at his university campus and wishes to locate a fax machine. This was implemented into an ontology containing 141 classes, 337 individuals and 126 roles. In the second case study Bob, in a foreign city centre, wants to find a movie cinema with an Internet café. The ontologies for this scenario contain 204 classes, 241 individuals and 93 roles.

We implemented the selective consistency, rank individuals, rank disjunctions and skip disjunctions strategies defined in the previous section, in the Pellet v1.5 reasoner. We selected Pellet because it is open source and implemented in java, allowing for easy portibility to small devices. Pellet supports OWL-DL with SHOIN expressivity. Table 1 presents a request for each case study and a positive/matching A and negative/non-matching B service individual, to be compared with each request.

Table 1. Type checks

Case	Request	Individual	Expected Result
1	Fax Laser Printer	A: #LaserPrinter1	Match
		B: #LaserPrinter2	No Match
2	Movie Cinema	A: #MovieCinema1	Match
		B: #MovieCinema2	No Match

Each of the four match checks in table 1 was executed 16 times using various randomly selected combinations of our optimisation strategies, as outlined in table 2. Test 16 represents normal execution of the Tableaux algorithm, with none of our optimisations strategies enabled.

Table 2. Optimisation tests

Test #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
C: Selective Consistency	✓	✓	✓	✓	✓	✓		✓	✓							
S: Skip Disjunctions	✓	✓	✓	✓			✓			✓	✓	✓				
D: Rank Disjunctions			✓	✓				✓	✓	✓	✓		✓	✓		
I: Rank Individuals		✓	✓			✓			✓	✓		✓	✓		✓	

We performed an evaluation on a HP iPAQ hx2700 PDA, with Intel PXA270 624Mhz processor, 64MB RAM, J2SE JVM, allocated 15MB of memory. Successfully executed tests returned the expected result shown in table 1. Figure 2 shows two graphs, which each show the consistency time to perform a type membership check for individual A and B against the request for the tests in table

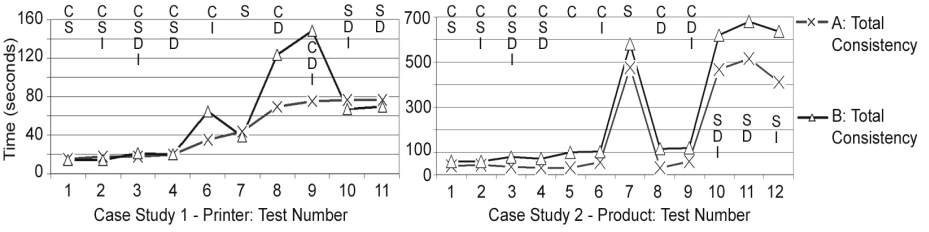

Fig. 2. Processing time taken to perform each test

2, for case study 1 and 2, respectively. Test 16, with no optimisations resulted in the "Out Of Memory" exception, necessitating our optimisations.

The figure shows that mTableaux optimisation can complete case study 1 and 2 in 18 and 35-70 seconds, respectively. This illustrates significant performance improvements in both scenarios. The selective consistency and skip disjunctions strategies were the most effective when used together. However, while we found that rank disjunctions and inviduals did reduce the number of branches applied, these did not significantly reduce reasoning time. Results also showed that the optimisations were less effective in improving performance for non-matching individuals B than matching individuals A, because the algorithm continues applying branches and completion rules until a clash is found.

Figure 3 illustrates the overhead cost incurred by each optimisation strategy.

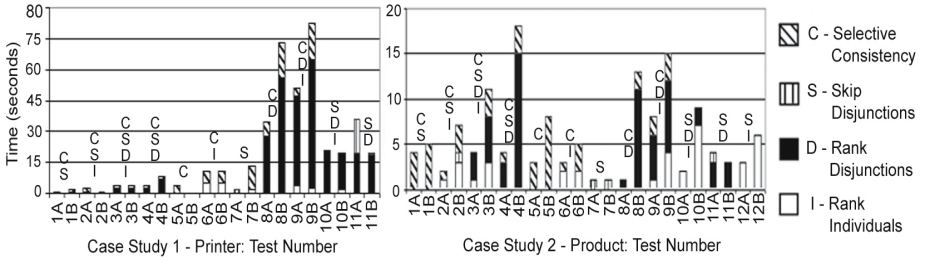

Fig. 3. Optimisation overhead breakdown. Each test was conducted for matching individual A and non-matching individual B.

We observed that selective consistency and skip disjunctions incurred low overhead, especially when used together. Rank disjunction overhead was significantly higher for tests 8 and 9 for both case studies due to the skip disjunction strategy being disabled, resulting in more disjunctions to evaluate. In addition to these results, we present a comparison between mTableaux, Pellet and Racer Pro in [11] which shows mTableaux out performs these, without reducing recall or precision.

4 Conclusion and Future Work

mTableaux was shown to significantly improve the performance of pervasive discovery reasoning tasks in two case studies, enabling them to be completed on small resource constrained devices. In future work we are leveraging our rank disjunctions and individuals strategies to adaptively reduce the number of branches applied when resources become low, to provide a less accurate result with a level of confidence to avoid "Out Of Memory" errors.

References

1. Arnold, K., O'Sullivan, B., Scheifler, R.W., Waldo, J., Woolrath, A.: The Jini Specification. Addison-Wesley, Reading (1999)
2. UPnP. Universal Plug and Play (UPnP), [cited March 12, 2007] (2007), `http://www.upnp.org`
3. Srinivasan, N., Paolucci, M., Sycara, K.: Semantic Web Service Discovery in the OWL-S IDE. In: 39th Hawaii International Conference on System Sciences, 2005, Hawaii (2005)
4. Küster, U., König-Ries, B., Klein, M.: Discovery and Mediation using DIANE Service Descriptions. In: Second Semantic Web Service Challenge 2006 Workshop, Budva, Montenegro, June 15-16 (2006)
5. FaCT++ [cited May 1, 2007] (2007), `http://owl.man.ac.uk/factplusplus`
6. RacerPro. [cited May 23, 2007] (2007), `http://www.racer-systems.com`
7. Pellet (2003), `http://www.mindswap.org/2003/pellet`
8. Horrocks, I., Sattler, U.: A Tableaux Decision Proceedure for SHOIQ. In: 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005), Morgan Kaufmann, San Francisco (2005)
9. Kleemann, T.: Towards Mobile Reasoning. In: International Workshop on Description Logics (DL 2006), Windermere, Lake District, UK, May 30 - June 1 (2006)
10. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
11. Steller, L., Krishnaswamy, S.: Pervasive Service Discovery: mTableaux Mobile Reasoning. In: International Conference on Semantic Systems (I-Semantics). Graz, Austria (2008)