Agent-based Matchmaking of Mathematical Web Services

Simone A. Ludwig and Omer F. Rana
School of Computer Science
Cardiff University
Cardiff, UK
{scmsal,scmofr}@cs.cardiff.ac.uk

William Naylor and Julian Padget
Department of Computer Science
University of Bath
Bath, UK
{wn,jap}@bath.ac.uk

ABSTRACT

Service discovery and matchmaking in a distributed environment has been an active research issue since at least the mid 1990s. Previous work on matchmaking has typically presented the problem and service descriptions as free or structured (marked-up) text, so that keyword searches, tree-matching or simple constraint solving are sufficient to identify matches. We discuss the problem of matchmaking for mathematical services, where the semantics play a critical role in determining the applicability or otherwise of a service and for which we use OpenMath descriptions of pre- and post-conditions. We describe a matchmaking architecture supporting the use of match plug-ins.

1. INTRODUCTION

The amount of machine-oriented data on the Web is increasing rapidly as semantic Web technologies achieve greater up-take. At the same time, the deployment of agent/Web Services is increasing and together create a problem for software agents that is the analog of the human user searching for the right HTML page. Humans typically use Google, but they can filter out the irrelevant and spot the useful, so while UDDI (the Web Services registry) with keyword searching essentially offers something similar, it is a long way from being very helpful. For many problems this is both appropriate and adequate, indeed it is not clear what more one could do, but in the particular domain of mathematical services the actual mathematical semantics are critical to determining the suitability (or otherwise) of the capability for the task. In the MONET (Mathematics on the NET) [2, 1] and GENSS (Grid-Enabled Numerical and Symbolic Services) [5] projects the objective is mathematical problem solving through service discovery and composition by means of intelligent brokerage. *Mathematical* capability descriptions turn out to be both a blessing and a curse: precise service descriptions are possible thanks to the use of the OpenMath [3] mathematical semantic mark-up, but service matching can rapidly turn into intractable (symbolic) mathematical calculations unless care is taken.

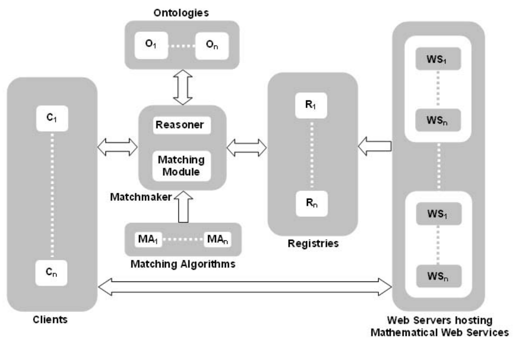

Figure 1: Matchmaking Architecture

2. MATHEMATICAL MATCHMAKING

2.1 Matchmaking Architecture

Our matchmaking architecture is shown in Figure 1 and comprises the following:

- Client interface: employed by the user to specify their service request.
- Matchmaker: contains the reasoning engine and matching module.
- Matching Algorithms: where the logic of the matching is defined.
- Mathematical ontologies: such as OpenMath Content Dictionaries (CDs), GAMS (General Algebraic Modeling System) etc.
- Registry service: where the mathematical service descriptions are stored.
- Mathematical Web Services: available on third party sites, accessible over the Web.

The interactions of a search request are as follows: (1) the user contacts the matchmaker, then (2) the matchmaker loads the matching algorithms specified by the user; in the case of an ontological match, further steps are necessary; (3) the matchmaker contacts the reasoner which in turn loads the corresponding ontology; (4) having additional match values results in the registry being queried, to see whether it contains services which match the request and finally (5) service details are returned to the user via the matchmaker.

The parameters stored in the registry (a database) are service name, URL, taxonomy, input, output, pre- and post-conditions. Using contact details of the service from the registry, the user can then invoke the mathematical Web Service.

2.2 Matching Algorithms

Currently four matching algorithms have been implemented within the matchmaker. These are structural match, syntax and ontological match, algebraic equivalence match and value substitution match. Service descriptions defined in OpenMath allow descriptions of mathematical pre- and post-conditions. The structural match only compares the OpenMath symbol element structures (e.g. `OMA`, `OMS`, `OMV` etc.). The syntax and ontological match algorithm goes a step further and compares the `OMS` elements `cd` and `name` attributes values. The algebraic equivalence match and value substitution match do actual mathematical reasoning using the OpenMath structure.

The **structural match** works as follows. The pre- and post-conditions are extracted and an SQL query is built to find the same OpenMath structure of the pre-/post- conditions of the service descriptions in the database.

The **ontological match** is performed similarly, however the OpenMath elements are compared with an ontology representing the OpenMath elements. The matchmaking mechanism allows a more effective matchmaking process by using mathematical ontologies. Let us assume that the part of the ontology given by the CD setname1 satisfies: $\mathbb{C} \supset \mathbb{R} \supset \mathbb{Q} \supset \mathbb{Z} \supset \mathbb{N} \supset \mathbb{P}$. If the user query contains the OpenMath element `<om:OMS cd='setname1' name='Z'/>` and the service description comprises `<om:OMS cd='setname1' name='P'/>`, then the query finds the entities `Z` and `P` and determines the similarity value depending on the distance between the two entities (inclusive, on one side) which in this case is $SV = \frac{1}{n} = 0.5$, where n is the degree of separation of the concepts. For both the ontological and structural match, it is necessary that the pre- and post-conditions are in some standard form. For instance, consider the algebraic expression $x^2 - y^2$, this could be represented in OpenMath as:

```
<om:OMOBJ><om:OMA>
  <om:OMS cd="arith1" name="minus"/>
  <om:OMA>
    <om:OMS cd="arith1" name="power"/>
    <om:OMV name="x"/>
    <om:OMI>2</om:OMI>
  </om:OMA>
  <om:OMA>
    <om:OMS cd="arith1" name="power"/>
    <om:OMV name="y"/>
    <om:OMI>2</om:OMI>
  </om:OMA></om:OMA>
</om:OMOBJ>
```

however, $x^2-y^2 = (x+y)(x-y)$, leading to ontologically and structurally different markup. Both are "right", it just depends on what information is wanted, so there can in general be no canonical form. In order to address the above observation, we must look deeper into the mathematical structure of the expressions which make up the conditions. Most of the conditions examined may be expressed in the form: Q(L(R)) where: Q is a quantifier block, e.g. $\forall x \exists y$ s.t. $\cdots$, L is a block of logical connectives, e.g. $\wedge, \vee, \Rightarrow, \cdots$ and R is a block of relations, e.g. $=, \leq, \geq, \neq, \cdots$.

In most cases, the quantifier block will just be a range restriction. Sometimes it may be possible to use *quantifier elimination* to replace the quantifier block by an augmented logical block. Once the quantifier elimination has been performed on the query descriptions and the service descriptions, the resulting logical blocks must be converted into normal forms. We now have a disjunction of terms which we are matching against a set of conjunctions of terms. It is useful to note that a term is of the general form: $T_L \succ T_R$ where $\succ$ is some relation i.e. a predicate on two arguments. In the case that T_L and T_R are real valued, we may proceed as follows: we have two terms we wish to compare $Q_L \succ Q_R$ and $S_L \succ S_R$, we first isolate an output variable r, this will give us terms $r \succ Q$ and $r \succ S$. There are two approaches which we now try in order to prove equivalence of $r \succ Q$ and $r \succ S$:

Algebraic equivalence — With this approach we try to show that the expression $(Q - S = 0)$ using algebraic means. There are many cases were this approach will work, however it has been proved [4] that in general this problem is undecidable. Another approach involves substitution of r determined from the condition $r \succ S$ into $r \succ Q$, and subsequently proving their equivalence.

Value substitution — With this approach we try to show that $(Q - S = 0)$ by substituting random values for each variable in the expression, then evaluating and checking to see if the valuation we get is zero. This is evidence that $(Q - S = 0)$, but is not conclusive, since we may have been unlucky in the case that the random values coincide with a zero of the expression.

3. CONCLUSION

We have presented an approach to matchmaking in the context of mathematical semantics. The additional semantic information greatly assists in identifying suitable services in some cases, but also significantly complicates matters in others, due to inherent richness of the service description. Consequently, we have put forward an extensible matchmaker architecture supporting plug-in matchers that may employ various reasoning techniques, utilising theorem provers and computer algebra systems as well as information retrieval from textual documentation of mathematical routines.

4. ACKNOWLEDGEMENTS

The work reported here is partially supported by the Engineering and Physical Sciences Research Council under the Semantic Grids call of the e-Science program (grant reference GR/S44723/01).

5. REFERENCES

[1] O. Caprotti et al. Mathematics on the (Semantic) Net. In *Proceedings of the European Symposium on the Semantic Web*, volume 3053 of *LNCS*, pages 213–224. Springer Verlag, 2004. May 2004.

[2] MONET Consortium. MONET Home Page, www. Available from `http://monet.nag.co.uk`.

[3] OpenMath Society. OpenMath website. `http://www.openmath.org`, February www.

[4] D. Richardson. Some Unsolvable Problems Involving Elementary Functions of a Real Variable. *Journal of Computational Logic*, 33:514–520, 1968.

[5] The GENSS Project. GENSS Home Page, www. Available from `http://genss.cs.bath.ac.uk`.