# Mathematical Matchmaker for Numeric and Symbolic 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 mathematical matchmaker supporting the use of match plug-ins.

## 1. INTRODUCTION

A significant number of applications within eScience make use of numerical algorithms, developed as part of the project or obtained from third parties such as numerical libraries from the Numerical Algorithms Group (NAG). The complexity of such algorithms can vary from simple matrix solving to more complex data analysis functions such as clustering or classification techniques. The ability to access such algorithms as Web Services allows easy integration of such capability within existing applications (while also providing a loose coupling between the application and the numerical algorithm). The matchmaker[1] developed focuses on the provision, discovery and use of mathematical services. The matchmaker assumes that in the future we are likely to see a service-rich environment where users will make available their mathematical libraries as services. The functionality being demonstrated will comprise of the matchmaker for mathematical services. Service descriptions are provided in an XML-based language (OpenMath [5]), which also allows descriptions of pre- and post-conditions. OpenMath has been adopted as it is being widely used by developers of mathematical libraries such as NAG.

[1] http://agentcities.cs.bath.ac.uk:8080/genss_axis/GENSSMatchmaker/index.htm


*AAMAS'05,* July 25-29, 2005, Utrecht, Netherlands.


## 2. MATHEMATICAL MATCHMAKER

The matchmaker developed consists of an architecture which supports five kinds of match modes, which include:

1. Basic structural match: this mode compares the XML tree structure of the OpenMath describing the query with that describing the service description (residing in the repository). OpenMath elements include OMA (OpenMath Application), OMS (OpenMath Symbol) this has attributes cd and name, OMV (OpenMath Variable) this has a name attribute, etc.

2. Syntax and ontology match: this mode first performs a structural match, and then performs a syntax match based on the attribute values of the OMS elements, and some inclusion rules referring to certain (set valued) OpenMath symbols.

3. Algebraic equivalence match: this mode compares the query with the service description by using algebraic means. Algebraic equivalence in this context is primarily related to how mathematical expressions can be re-written.

4. Value substitution match: this mode compares the query and a service, and substitutes suitable values into each to check for "value equivalence". Essentially, this mode of comparison does not guarantee that the two expression are similar, only that for the test data being used for evaluation, both return the same results.

5. Decomposition match: in case an exact service match cannot be found, the decomposition match will attempt to discover an equivalent mathematical expression. Essentially, this involves dividing the query into sub-queries, and trying to find a match for each decomposed sub-query. The decomposition is supported by applying a set of rules that try to match each service description. The rules are applied recursively to decompose a mathematical expression into its simplest form.

The matchmaker uses the individual match scores from each of the plug-ins above to compute a ranking of matched services.

The matchmaker[2] consists of two menus which are:

- List services

[2] http://agentcities.cs.bath.ac.uk:8080/genss_axis/demo/GENSS.avi

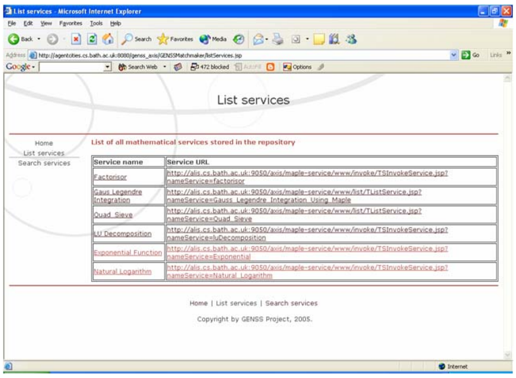


**Figure 1: List Services**

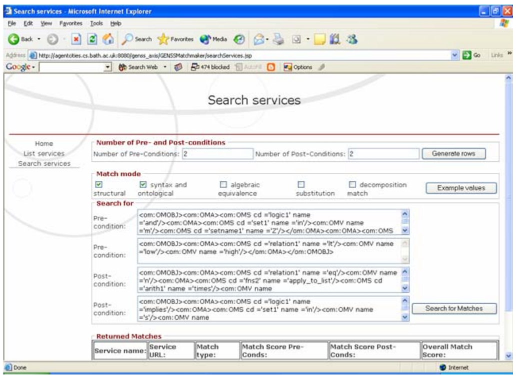


**Figure 2: Search Services (1)**

- Search services

The *List services* menu, shown in Figure 1, lists all mathematical services stored in the repository displaying service name and service URL. By clicking on the service name another screen opens displaying all the details about the service such as service name, service URL, and pre- and post-conditions in OpenMath syntax. By clicking on the service URL another screen pops up allowing to invoke the mathematical service.

Menu *Search services*, shown in Figure 2, allows the user to search for mathematical services defining queries in OpenMath syntax. This screen contains a section where the number of pre- and post-conditions can be given in order to provide the input fields for the pre- and post-conditions. The match mode section provides the five matching modes (structural, ontological, algebraic equivalence, substitution and decomposition). The *Search for Matches* button performs the matchmaking depending on the match mode chosen. The results of matched services, shown in Figure 3, are displayed in the table *Returned Matches* showing service details such as service name, service URL and match type plus three match score numbers for pre-, post-conditions and overall respectively.

## 3. IMPLEMENTATION

This prototype system is based on latest web services technology standards. It consists of the following components.

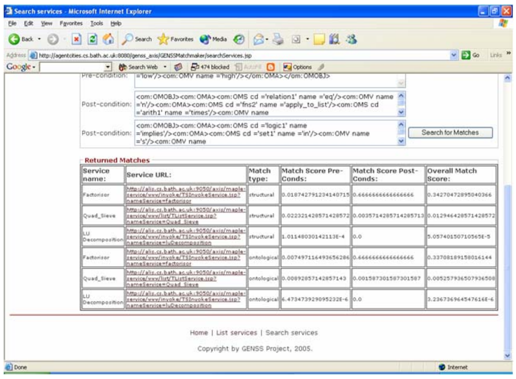


**Figure 3: Search Services (2)**

The *Client* is build as a web client using Java Server Pages (JSPs) [1]. The *matchmaking service* is implemented as a Web service using WSDL (Web Service Description Language) and SOAP (Simple Object Access Protocol) [4]. The *matchmaking algorithms* are also implemented as web services and are registered in a *UDDI* [4] registry. This allows the matchmaking service to dynamically load the different matchmaking algorithms specified. The *service registry* is implemented as a database to store the mathematical service descriptions using MySQL [2]. The *ontology service* consists of an OWL ontology and a JESS (Java Expert Systems Shell) [3] engine as a reasoner. JESS was chosen as a rule-based language for the matchmaker as it provides the functionality for defining rules and queries in order to reason about the ontologies specified. The *Mathematical Web services* provide the numerical and symbolic services.

## 4. CONCLUSION

With the matchmaker implementation 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 their inherent richness. Consequently, we have put forward an extensible matchmaker architecture supporting plug-in matchers that may employ a variety of reasoning techniques, utilising theorem provers and computer algebra systems as well as information retrieval from textual documentation of mathematical routines.

## 5. REFERENCES

[1] J2EE JavaServer Pages Technology. Available from `http://java.sun.com/products/jsp/`.

[2] Java Expert Systems Shell. Available from `http://www.mysql.com/`.

[3] Java Expert Systems Shell. Available from `http://herzberg.ca.sandia.gov/jess/docs/61/index.html`.

[4] A.E. Walsh. UDDI, SOAP, and WSDL: The Web Services Specification Reference Book, 2002. UDDIORG.

[5] The OpenMath Society. The OpenMath Standard, October 2002. Available from `http://www.openmath.org/standard/om11/omstd11.xml`.