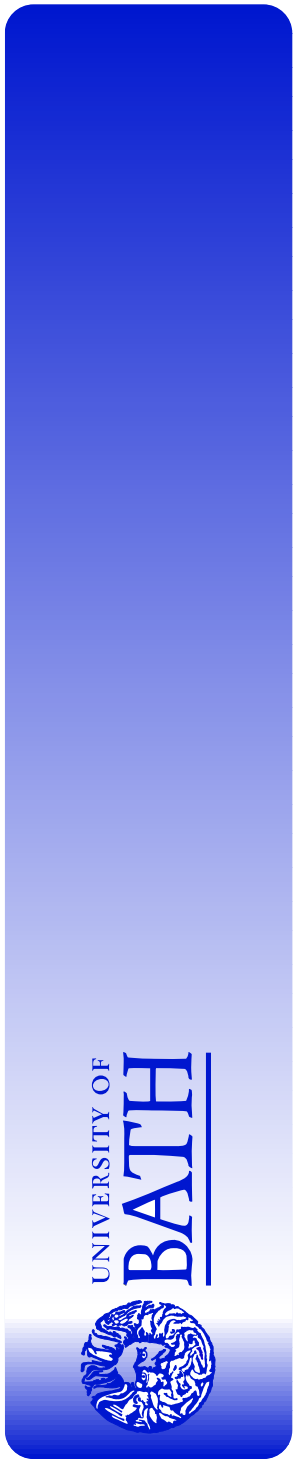


Matchmaking Support for Mathematical Web Services

Simone Ludwig[1], William Naylor[2], Julian Padget[2], Omer Rana[1]

[1]Department of Computer Science, Cardiff University, UK
{scmsal,scmofr}@cs.cardiff.ac.uk
[2]Department of Computer Science, University of Bath, Bath, UK
{wn,jap}@cs.bath.ac.uk
http://genss.cs.bath.ac.uk

Presented by Julian Padget

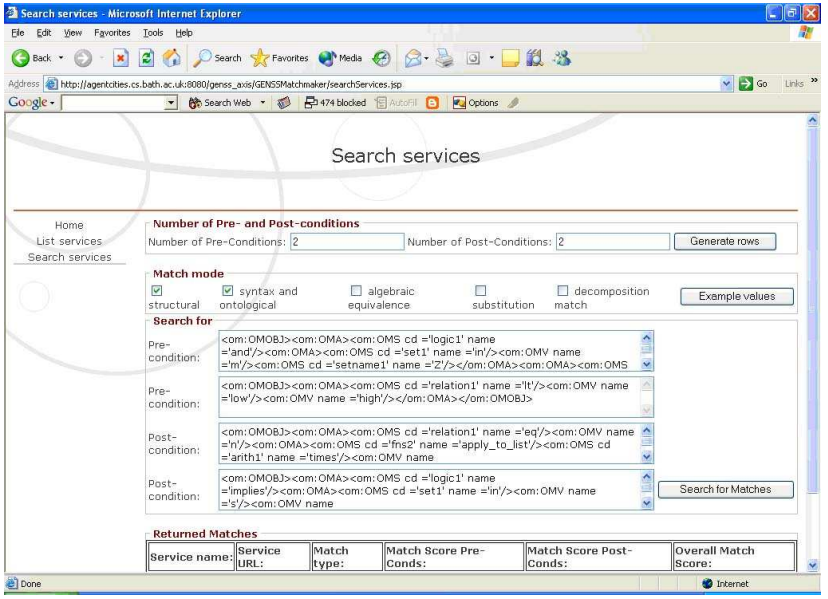
Outline

- Goals and progress
- Background
- Matchmaking and normalization
- Similarity computation
- Service composition
- Outputs

The Vision

Identification and composition of mathematical web services by functional description:

e-scientist

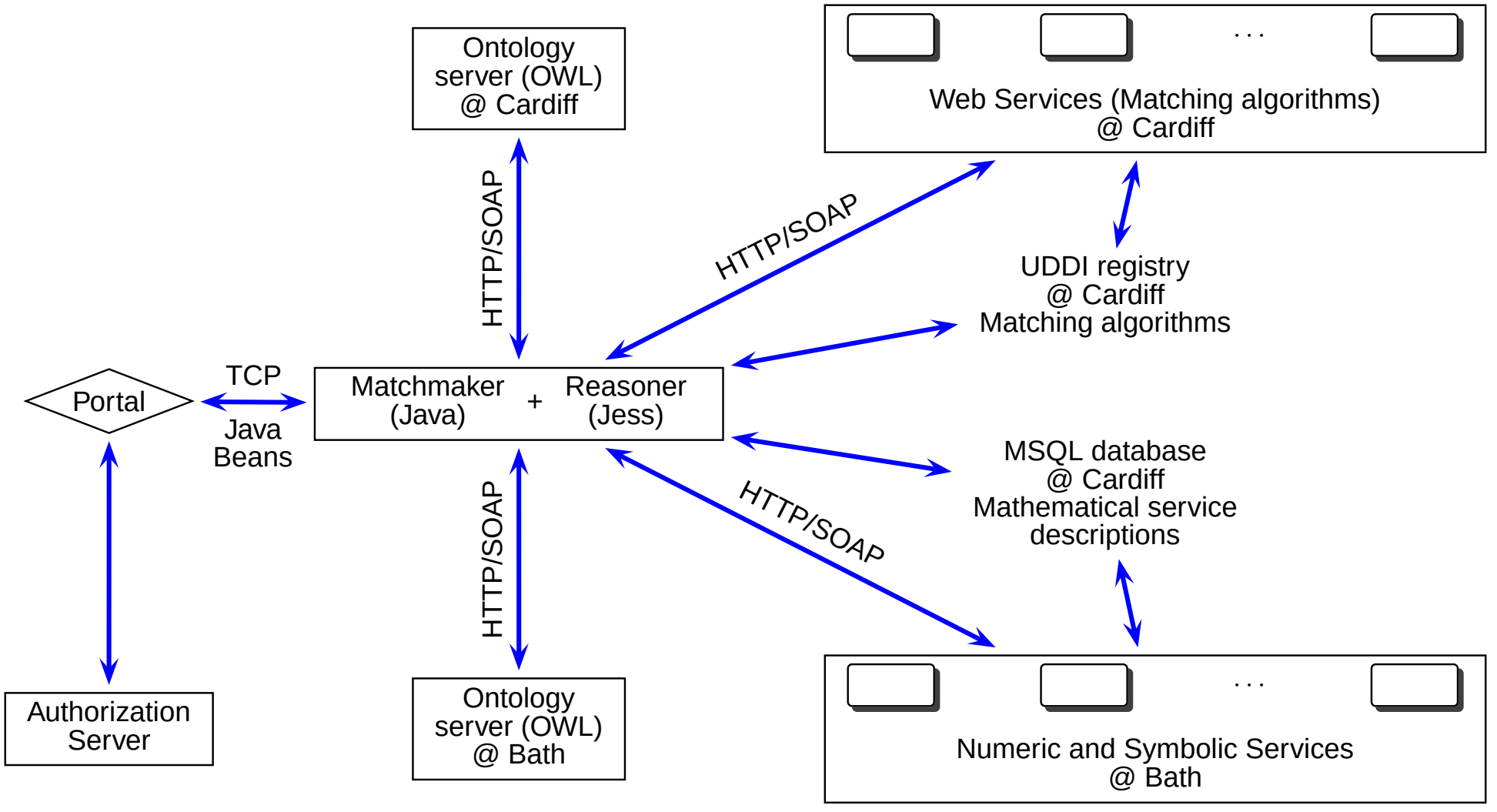


mathematical web services

software agent

Plumbing is easy, saying what you want is not

The Reality


Ontology server (OWL) @ Cardiff
Web Services (Matching algorithms) @ Cardiff
HTTP/SOAP
HTTP/SOAP
UDDI registry @ Cardiff Matching algorithms
Portal
TCP
Java Beans
Matchmaker (Java) + Reasoner (Jess)
MSQL database @ Cardiff Mathematical service descriptions
HTTP/SOAP
HTTP/SOAP
Authorization Server
Ontology server (OWL) @ Bath
Numeric and Symbolic Services @ Bath

From MONET to GENSS

- MONET: http://monet.nag.co.uk, EU, 2002–2004
  - Schema definition: Mathematical Service Description Language, Mathematical Problem Description Language, Mathematical Explanation Language
  - Basic ontologies: hardware, software, algorithms, problems
  - OpenMath: http://www.openmath.org, mathematical symbols, concepts, organized by content dictionaries (CDs)
  - Brokerage, service selection by ontological reasoning, orchestration with BPEL4WS
- GENSS: http://genss.cs.bath.ac.uk, EPSRC, 2004–2006
  - Extend matching capability to capabilities and effects
  - Diversify range of services
  - Integrate with grid

GENSS Matchmaking Strategy

Two stages:

- Service registration:
  1) Convert service to normal form
  2) Put in registry

  Conversion to normal form dominates complexity.
- Service lookup:
  1) Convert query to normal form
  2) Traverse registry, calculating a similarity value between the query and each service
  3) Return a list of service URLs ordered on the similarity value.

  Traversal of the registry dominates complexity.

Matchmaking framework supports plug-ins implemented as web services

Normalization

- Dissimilar (mathematical) expressions can be equivalent
- No absolute normal form exists
- But can normalize for purpose:
  - Logical equivalences – standard rewrites
  - Associative operators – flattened
  - Context dependent equivalences
  - Alpha conversions – consistent naming
  - Commutative operators – reorder arguments
  - Conversion to disjunctive normal form (cost: $O(2^n)$)
- Hence, pre- and post-conditions take the form Q(L(R)), where:
  - Q is a quantifier block e.g. $\forall x \exists y \text{ s.t.} \cdots$
  - L is a block of logical connectives e.g. $\wedge, \vee, \Rightarrow, \cdots$
  - R is a block of relations. e.g. $=, \leq, \geq, \neq, \cdots$

Matching techniques

- Structural: when task and capability match exactly
- Syntax+Ontology:
  - Compare elements and attributes in task and capability using taxonomic structure of types to test inclusion
  - The MONET matcher (to be incorporated)
    - Sort capabilities that satisfy: $T_{in} \geq C_{in} \wedge T_{out} \leq C_{out}$
    - Compute match using DL reasoner
- Function:
  - Use pre/post-conditions: $T_{pre} \Rightarrow C_{pre} \wedge C_{post} \Rightarrow T_{post}$
    - *Algebraic equivalence*: show that $Q - S = 0$ algebraically. In general undecidable, but often works in practice. eg. $x^2 - y^2$ and $(x + y)(x - y)$.
    - *Value substitution*: show that $Q - S = 0$ by substituting values into the expression and evaluating. Result is *evidence*, not *proof*.

Task:capability similarity 1/3

UNIVERSITY OF BATH

- The similarity measure computation starts from

$$T_{pre} \Rightarrow C_{pre} \wedge C_{post} \Rightarrow T_{post}$$

- Considering the DNF of T_{pre}, T_{post}, C_{pre} and C_{post} as sets of conjuncts, this says:

  There exists at least one conjunct in C_{pre} (T_{post}) which is implied by every conjunct in T_{pre} (C_{post})

- More formally the condition is equivalent to:

$$\exists x_1 \in C_{pre} \text{ s.t. } \forall y_1 \in T_{pre} \mid y_1 \Rightarrow x_1$$

and

$$\exists x_2 \in T_{post} \text{ s.t. } \forall y_2 \in C_{post} \mid y_2 \Rightarrow x_2$$

Task:capability similarity 2/3

- Given two DNF expressions R and S:

$$\text{similarity} = \max_{i=1..n}\{\min_{j=1..\tilde{n}}\{M(R_i, S_j)\}\}$$

- $M(R_i, S_j)$ calculates a similarity value between two conjuncts (of terms)
- **min** ranges over the *supplier* (T_{pre} or C_{post}) as every conjunct is required, *so min is as good as it gets*
- **max** ranges over the *requirer* (C_{pre} or T_{post}) as any conjunct is possible, *so take the best*

Task:capability similarity 3/3

- General formula for $M(R_i, S_j)$

$$M(R_i, S_j) = \sum_{k=1 \cdot\cdot \delta} m(S_j, R_{i,k}) w_k$$

where $R_{i,k}$ are the terms in R_i,
w_k is a weighting value
and $m(S_j, R_{i,k})$ is a comparison function – algebraic equivalence or value substitution.

- We weight all terms equally, so take $w_k = \frac{1}{\delta}$
- $M(S_j, R_{i,k})$ calculates the similarity value of terms and conjuncts.

Composition of services

- Task: T_{pre}, T_{post}
  Capabilities: C_{pre_1}, C_{post_1} and C_{pre_2}, C_{post_2}
- Conditions for satisfiability:

$$\begin{aligned} T_{pre} &\Rightarrow C_{pre_1} \\ T_{pre} \wedge C_{post_1} &\Rightarrow C_{pre_2} \\ C_{post_1} \wedge C_{post_2} &\Rightarrow T_{post} \end{aligned}$$

Current and future work

- Deomonstrate integration with grid services
- Combinatorial Auction and Answer Set solvers as mathematical services
- Release of matchmaker components: plug-ins as web services
- Publication of a repository of mathematical service descriptions
- Demonstration of static and dynamic service composition derived from functional properties of services

[Project ends Spring 2006]