The MathServe System for Semantic Web Reasoning Services

Jürgen Zimmer[1] and Serge Autexier[1,2]

[1] FB Informatik, Universität des Saarlandes, D-66041 Saarbrücken, Germany.
[2] DFKI, Stuhlsatzenhausweg 3, D-66123 Saarbrücken, Germany,
{jzimmer|autexier}@ags.uni-sb.de

1 Introduction

In recent years, formal verification of hardware and software components has increasingly attracted interest from both academia and industry. The widespread use of automated reasoning techniques requires tools that are easy to use and support standardised protocols and data exchange formats. In [1] the first author presented the MathWeb Software Bus, a first step towards re-usable reasoning services. The MathWeb-SB had several drawbacks which limited its usability. For example, it had no service brokering capabilities and *the user* had to know exactly which reasoning system to use to solve a problem and how to access it.

Here we present the MathServe system that overcomes the limitations of the MathWeb-SB. MathServe offers reasoning systems as Semantic Web Services described in OWL-S [2]. MathServe's service broker can automatically find suitable services (and compositions of services) to solve a given reasoning problem. The use of Semantic Web technology allows applications and humans to automatically retrieve and access reasoning services via the Internet. MathServe complements similar projects, such as the *MathBroker* project [3], which describe computational services as Semantic Web Services. We provide and overview of the MathServe system in § 2 and describe the evaluation of the system at CASC-20 in § 3. We conclude and discuss future work in § 4.

2 The MathServe System

The MathServe system is based on state-of-the-art technologies for Semantic Web Services: It integrates reasoning systems as *Web Services*. The semantics of these Web Services is described in the OWL-S upper ontology for Web Services [2]. OWL-S *service profiles* define the inputs and outputs as well as the preconditions and effects of Web Services. MathServe services and the service broker are accessed by means of standard Web Service languages and protocols.

Client applications can interact with MathServe in two principal ways: All reasoning services can be invoked individually with reasoning problems. Complex queries can be sent to the MathServe broker which can perform service matchmaking and automated service composition. Given a query containing a reasoning problem, service matchmaking returns a list of standalone services

that can potentially answer that query. So far, service matchmaking simply performs class subsumption tests on the types of input and output parameters of available services and the query provided. If no single service can answer a query, MathServe's *service composer* can automatically combine services using classical AI planning and decision-theoretic reasoning.

Reasoning problems and their solutions are encoded in OWL/RDF format [4]. The primary interface to MathServe is the standard Web Service interface defined in the Web Service Description Language (WSDL). Services are invoked via the Simple Object Access Protocol (SOAP). Tools and libraries for WSDL and SOAP are available in many mainstream programming languages. Next to the SOAP interface, the MathServe broker offers an XML-RPC interface with convenient interface methods similar to the ones described in [1].

Reasoning Services in MathServe. MathServe provides several reasoning systems for classical first-order logic with equality as Semantic Web Services: 1) Services for clause normal form transformation of first-order problems are provided by the tptp2X utility and the FLOTTER system (available with SPASS [5]). 2) A problem analysing service can determine the *Specialist Problem Class* of a theorem proving problem (see below). 3) Deduction services are provided by state-of-the-art Automated Theorem Proving (ATP) systems. 4) Transformations of formal proofs are performed by the systems Otterfier [6] and TRAMP [7]. We cannot describe all these services in this paper. Detailed descriptions of the services provided by Otterfier and TRAMP can be found in [6]. In this paper, we focus on first-order ATP services. With respect to ATP services, MathServe is similar to the SSCPA system[3], which has been developed for human users, while MathServe's services can be consumed by software applications.

The latest version of MathServe offers the ATP systems DCTP 10.21p, EP 0.91, Otter 3.3, SPASS 2.2, Vampire 8.0 and Waldmeister 704 as Semantic Web Services (see [5] for ATP system descriptions). All theorem proving services support the same Web Service interface and can be invoked with a theorem proving problem, a CPU time limit, and (optionally) prover-specific options. The answers provided by ATP services specify unambiguously what has been established by the underlying system. For this, we developed an ontology of 18 well-defined ATP statuses [8]. Furthermore, results of ATP services contain the complete output of the prover, a reference to the problem submitted, the CPU and wall-clock time used, and, if available, resolution proofs in the new TPTP format.

OWL-S Descriptions of ATP Services. The performance of an ATP system depends on the computational resources given to the prover as well as the type of the problem to solve. *Sutcliffe* and *Suttner* [9] identified six "objective problem features" that have an impact on the performance of ATP systems. The meaningful combinations of these features define 21 *Specialist Problem Classes*

[3] Accessible via the TPTP web site (`http://www.tptp.org`).

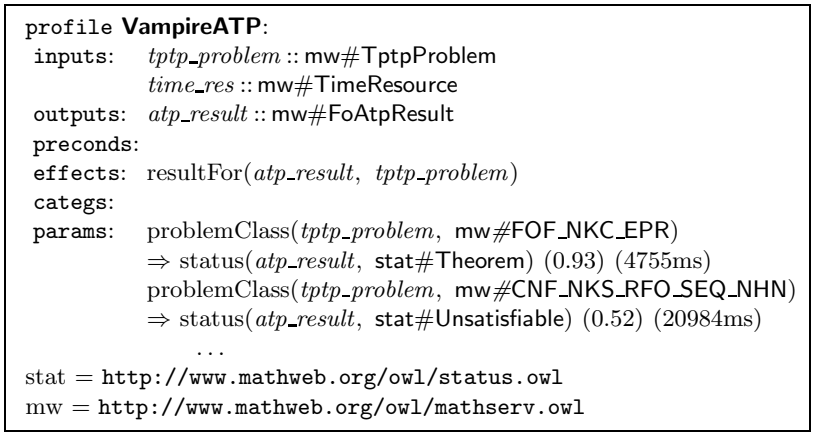

Fig. 1. The OWL-S service profile of VampireATP

(SPCs) [9]. [4] All OWL-S service profiles for the above-mentioned ATP systems are annotated with data reflecting the performance of the system on the TPTP Library v3.1.1. For every SPC and every ATP system we have calculated the ratio of (TPTP Library) problems in that SPC solved by the system. We assume this ratio to be a good estimate for the system's probability of success on that SPC. The average CPU time for solved problems represents the average cost of the ATP service on that SPC. This performance data is modelled as conditional probabilistic effects of a service profile and is used by the MathServe broker to choose the most suitable ATP service for given proving problems.

Fig. 1 shows the OWL-S service profile of the service for Vampire 8.0.[5] Like all ATP services VampireATP takes a problem in the new TPTP format and a TimeResouce as inputs. It returns a first-order ATP result as described above. The *service parameters* contain the performance information, which indicates, for instance, that the service can prove the input problem with probability 93% if the problem is in the SPC FOF_NKC_EPR. The average time for proving conjectures in this SPC is 4.8 secs. For CNF_NKS_RFO_SEQ_NHN the probability of success is only 52% and the average CPU time for proving is 21 secs. The service profile contains similar statements for the remaining 19 SPCs.

Automated Reasoning Service Composition. Service composition in MathServe is a two-stage process. In the first stage the classical AI planning system PRODIGY [10] is used to find suitable sequences of reasoning services that can potentially answer a given query. In the second stage, the plans found by PRODIGY are combined and the probabilistic effects of a service (e.g., the per-

[4] For instance, CNF_NKS_RFO_SEQ_NHN is the SPC of problems in clause normal form that are potential "theorems" (not known to be satisfiable, NKS), are real first-order problems (RFO), contain some equality literals (SEQ) and non-Horn clauses (NHN).

[5] The XML namespaces stat and mw refer to MathServe's domain ontologies.

formance data in Fig. 1) are taken into account. A decision-theoretic reasoner computes an *optimal policy*, i.e. a program with conditional statements that maximises the probability of success by choosing different (parts of) plans in the context of a particular reasoning problem. In the case of ATP services, for instance, the optimal policy first analyses the problem at hand and then chooses the most promising ATP service depending on the SPC of the problem.

3 System Evaluation

MathServe participated in the demonstration division of the 20th CADE ATP System Competition (CASC-20) [5]. We evaluated MathServe's brokering capabilities for theorem proving services. Since MathServe does not constitute a new ATP system, but employs other ATP systems, it participated in the *demonstration division* in which systems are not formally ranked. As a preparation for the system competition we measured the performance of the ATP systems EP 0.82, Otter 3.3, SPASS 2.1 and Vampire 7.0 on the TPTP Library (v3.0.1) with a CPU time limit of 300 seconds per problem. The OWL-S profiles were annotated with the resulting performance information as described in the previous section. The MathServe broker computed an optimal policy from this data. The problem set of CASC-20 was composed of 660 randomly chosen problems from the TPTP Library (v3.1.0). 147 of these problems had not been seen by the competition participants before. A MathServe client was run sequentially on all 660 problems with a 600sec CPU time limit. The broker's optimal policy determined the SPC of each problem and chose the most promising ATP service according to the performance data recorded before. MathServe could solve 392 problems. Table 1 shows that MathServe could not solve as many problems as the leading ATP systems EP (409) and Vampire (430). This was due to the significant improvements made to the most recent versions of these ATP systems. These improved systems were not available to MathServe at the time of the competition. Furthermore, MathServe could not handle six problems of extraordinary size (> 2MB). If MathServe had used the competition versions of EP and Vampire (and the corresponding optimal policy) it would have solved 440 problems. After CASC-20 we improved MathServe by integrating the latest versions of the ATP systems EP and Vampire as well as the specialised provers

Table 1. MathServe's performance at CASC-20 compared to EP and Vampire

System	Problems given	Problems solved	Percentage of given	Percentage complete
MathServe 0.62	660	392	59.4%	59.4%
Ep 0.9pre3	660	409	62.0%	62.0%
Vampire 8.0	540	430	79.6%	65.2%

DCTP and Waldmeister. We also changed the problem processing of MathServe such that it can now handle large problems.

4 Conclusions and Future Work

We described the MathServe system which offers several reasoning systems as Semantic Web Services. All services are accessible via standard Web Service protocols. MathServe's service broker can automatically find suitable services (and service compositions) for a given reasoning problem. The evaluation of MathServe in CASC-20 showed that the system performs well in a competition environment. In the future we will extend MathServe with services provided by finite model finding systems and decision procedures. Furthermore, we are planning to enhance the semantic descriptions of composite services by allowing parallel execution (as offered by the SCCPA system) and explicit time resource assignments.

The MathServe system is free software available under the GNU Public License. A binary distribution of the MathServe system is available from the system web page at `http://www.ags.uni-sb.de/~jzimmer/mathserve.html`. The system sources can be obtained via anonymous CVS.

References

1. Zimmer, J., Kohlhase, M.: System Description: The Mathweb Software Bus for Distributed Mathematical Reasoning. In Voronkov, A., ed.: Proc. of CADE-18. Number 2392 in LNAI, Springer Verlag, Berlin (2002) 139–143
2. Martin, D., et al.: OWL-S: Semantic Markup for Web Services. `http://www.daml.org/services/owl-s/1.1/overview` (2004)
3. Schreiner, W., Caprotti, O.: The MathBroker Project. `http://poseidon.risc.uni-linz.ac.at:8080/mathbroker/` (2001)
4. Bechhofer, S., van Harmelen, F., Hendler, F.U.A.J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P.F., Stein, L.A.: OWL Web Ontology Language Reference. `http://www.w3.org/TR/owl-ref/` (2004)
5. Sutcliffe, G.: The CADE-20 Automated Theorem Proving Competition. AI Communications (2006)
6. Zimmer, J., Meier, A., Sutcliffe, G., Zhang, Y.: Integrated Proof Transformation Services. In Benzmüller, C., Windsteiger, W., eds.: Proc. of the Workshop on Computer-Supported Mathematical Theory Development, Cork, Ireland (2004)
7. Meier, A.: TRAMP: Transformation of Machine-Found Proofs into Natural Deduction Proofs at the Assertion Level. In McAllester, D., ed.: Proc. of CADE-17. Volume 1831 of LNAI., Springer Verlag (2000) 460–464
8. Sutcliffe, G., Zimmer, J., Schulz, S.: Communication Fomalisms for Automated Theorem Proving Tools. In Sorge, V., Colton, S., Fisher, M., Gow, J., eds.: Proc. of the IJCAI'03 Workshop on Agents and Automated Reasoning. (2003)
9. Sutcliffe, G., Suttner, C.: Evaluating General Purpose Automated Theorem Proving Systems. Artificial Intelligence **131**(1-2) (2001) 39–54
10. Veloso, M., et al.: Integrating Planning and Learning: The PRODIGY Architecture. Journal of Experimental and Theoretical Artificial Intelligence **7**(1) (1995)