SPASS-XDB goes Mathematical

David Stanovský[1]*, Martin Suda[2], and Geoff Sutcliffe[3]

[1] Charles University in Prague, Czech Republic
[2] Max-Planck-Institut für Informatik, Germany
[3] University of Miami, USA

1 Introduction

There is a growing demand for automated reasoning with *world knowledge* [2]. For example, a reasoning system with knowledge of what cities are located near water, and access to data describing current weather conditions, would be able to predict which cities might be flooded (e.g., by tidal surge). World knowledge is available from a growing number of sources, e.g., online databases, SPARQL endpoints, web services, and computational systems. For deep reasoning, world knowledge is used in conjunction with ontological axioms that describe the structure of the world in which the knowledge resides. Automated Theorem Proving (ATP) systems have traditionally not been well suited to reasoning with world knowledge, because they expect to load all the formulae before deduction starts. The large (possibly infinite) number of axioms available from world knowledge sources dictates that the axioms be retrieved dynamically during deduction.

Mathematics is an established facet of automated reasoning [1], and is often required when reasoning with world knowledge. The requirement is often computational, e.g., evaluating ground arithmetic expressions, but can be more abstract. ATP systems have traditionally lacked arithmetic capabilities, because arithmetic cannot be finitely axiomatized, and building in arithmetic has not meshed well with reasoning over uninterpreted symbols. Recent developments have started to overcome this weakness – the TPTP [4] was extended to support the Typed First-order Form (TFF) language, which provides the basis for the Typed First-order with Arithmetic (TFA) part of the TPTP.

SPASS-XDB [3] is an ATP system that incorporates world knowledge axioms from multiple external sources, asynchronously on demand, during its deduction process. SPASS-XDB includes internal support for evaluation of ground arithmetic expressions, and uses Mathematica as an external source of mathematical knowledge. This paper describes SPASS-XDB.

2 The SPASS-XDB Architecture

SPASS-XDB is a modified version of the well-known, state-of-the-art, first-order ATP system SPASS [6]. The complete system architecture is comprised of the SPASS-XDB system itself, mediators, and external sources of world knowledge

* Supported by MSM 0021620839 and GAČR 201/08/P056.

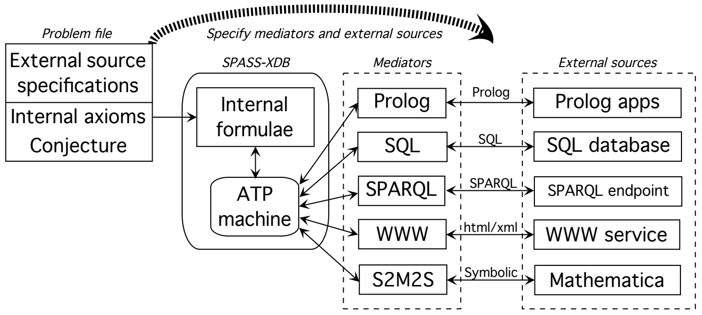

Fig. 1. System Architecture

axioms, as shown in Figure 1. The problem file, including the external source specifications, is written in the TPTP language. The external sources currently supported include DBPedia, the Linked Movie Database, Mathematica (see Section 3), Mondial, and large fragments of the YAGO database. The addition of new external sources is quite simple. SPASS-XDB augments SPASS' classic CNF saturation algorithm with steps to request and accept axioms from the external sources. Requests are made when a negative literal of the "given" clause (of SPASS-XDB's saturation loop) matches the template of an external specification. A request is sent in TPTP format to the corresponding mediator, which marshals the request into a query for the external source, retrieves matching facts from the external source, unmarshals the facts into TPTP format axioms, and delivers them to SPASS-XDB. The requests are made and axioms delivered asynchronously, so that SPASS-XDB continues its deduction process while axioms are being retrieved. The axioms are integrated into the deduction process by adding them to SPASS-XDB's "usable" list. SPASS-XDB additionally implements new inference rules that provide arithmetic reasoning, described in Section 3.

3 SPASS-XDB goes Mathematical

The TPTP language provides constructs for writing Typed First-order with Arithmetic (TFA) problems.[4] Separate signatures are assumed for integer, rational, and real arithmetic, each comprised of infinitely many unequal numeric constants. A suite of arithmetic function and predicate symbols are defined, as shown in Table 1 (with their Mathematica equivalents – see below). For SPASS-XDB, TFA problems are translated to First-Order Form (FOF) using a standard translation, converting types to monadic type predicates, using the type check predicates of Table 1 for numeric variables. SPASS-XDB solves these problems using its regular inferencing, some new inference rules for arithmetic, and external arithmetic axioms provided by Mathematica.

Numbers are represented in SPASS-XDB as special constant symbols that carry both the type and the value. The GMP arithmetic library is used for arbitrary precision computation. The basic mathematical functionality is provided

[4] `http://www.tptp.org/TPTP/TR/TPTPTR.shtml#Arithmetic`

Operation	TPTP predicates and functions				
	Mathematica operators				
Comparison	`=`	`$less`	`$lesseq`	`$greater`	`$greatereq`
	==	<	<=	>	>=
Computation	`$uminus`	`$sum`	`$difference`	`$product`	
	−	+	−	∗	
Coincidence	`$is_int`		`$is_rat`		
	Element[X,Integers]		. . . Rationals		
Coercion	`$to_int`	`$to_rat`	`$to_real`		
	Floor[X]	X	X		
Type check	`$int`		`$rat`	`$real`	
	Element[X,Integers]		. . . Rationals	. . . Reals	

Table 1. TPTP and Mathematica Arithmetic

by a new inference rule called *ground arithmetic rewriting*. Given a clause as a premise, it traverses the term structure of all its literals in a bottom up fashion, and performs the symbolically represented arithmetic operations whenever all arguments are numbers (thus even terms below uninterpreted symbols may get simplified). Arithmetic predicates (including equality) may get evaluated, which simplifies the clause, or might show its redundancy.

To extend the mathematical capabilities of SPASS-XDB beyond ground arithmetic rewriting, Mathematica is used as an external source of axioms. SPASS-XDB generates requests from negative arithmetic literals (see below for how the one-literal-at-a-time approach is extended to multiple-literals-at-a-time in requests for arithmetic knowledge). The S2M2S mediator translates a request from SPASS-XDB into the Mathematica language, following the interpretation given in Table 1, and calls the `FindInstance` function of Mathematica. `FindInstance` inputs an expression and a set of variables that occur in the expression, and finds an instance of the variables that makes the statement true. Multiple answers can be demanded and a time limit can be imposed. The mediator reads the answer, and creates new axioms that are passed back to SPASS-XDB.

External sources of mathematical knowledge have several special features: semantically, the polarity of arithmetical literals is undefined (e.g., $\neg(x < y)$ iff $x \geq y$); there is a potentially infinite language to describe the answer; there can be infinitely many answers to a single query. To solve the *polarity* issue, an additional inference rule has been implemented in SPASS-XDB, which converts all syntactically positive arithmetic literals to negative ones, e.g., $x < y$ is converted to $\neg(x \geq y)$. Positive equations are converted to negated inequations, which are represented using a new internal predicate symbol `$$arith_uneq`. A complementary inference rule is applied to a clause to convert negated inequations back to positive ones so that they can take part in superposition inferences. The *potentially infinite language* manifests itself (at least) when irrational solutions are found, such as for the request $\exists X.[(\$\mathrm{real}(X) \wedge (X{*}X = 2)]$. Mathematica returns a precise value (e.g., $\sqrt{2}$), in its internal language (e.g., `Sqrt[2]`). The answers are passed to SPASS-XDB using a new internal function symbol `$$mathematica`, with a string argument that contains the Mathematica expression describing the number (in the example, `$$mathematica("Sqrt[2]")`). Such values can be used in further calls to Mathematica, e.g., `$product($$mathematica("Sqrt[2]"), $$mathematica("Sqrt[2]"))` is evaluated by Mathematica to 2.

To reduce the *number of solutions*, multi-literal queries are used. The existing SPASS-XDB implementation built requests using only a single negative literal of the given clause. For problems with arithmetic, SPASS-XDB takes advantage of the fact that Mathematica understands all arithmetic and logical symbols, and can build requests from multiple literals of the given clause. When SPASS-XDB selects a negative literal that contains arithmetic symbols, all negative literals in the clause are scanned to check whether they can be conjoined with the selected literal into a single request. This often reduces the number of solutions (i.e., axioms that could be delivered to SPASS-XDB) from infinitely many to very few. For example, consider the following clause:

$$\neg\$\mathrm{int}(X) \ \vee\ \neg\$\mathrm{int}(Y) \ \vee\ \neg(X \leq 0) \ \vee\ \neg(X * X = 10000) \ \vee\ p(X, Y)$$

Assume the literal $\neg(X \leq 0)$ is selected. SPASS-XDB detects that the literals $\neg\$\mathrm{int}(X)$ and $\neg(X * X = 10000)$ can be conjoined with it to form the request $\exists X.[(X \leq 0) \ \wedge\ \$\mathrm{int}(X) \ \wedge\ (X * X = 10000)]$, and Mathematica computes the single solution -100. Without multi-literal querying there would be infinitely many answers to a request based on only the first literal, and the chance of obtaining a solution that matches the other literals would be very low. For multi-literal requests, axioms that are instances of the first literal in the request are returned, e.g., $(-100 \leq 0)$. Such axioms resolve with the original clause, e.g., resulting in the clause

$$\neg\$\mathrm{int}(-100) \ \vee\ \neg\$\mathrm{int}(Y) \ \vee\ \neg((-100) * (-100) = 10000) \ \vee\ p(-100, Y)$$

which is reduced to $\neg\$\mathrm{int}(Y) \ \vee\ p(-100, Y)$ by ground arithmetic rewriting. Note that the multi-literal request capability is necessary to include the type predicates even in cases based on a single literal.

4 Performance

SPASS-XDB and other TFA-capable ATP systems have been tested on the 875 TFA problems in TPTP v5.1.0. All the other systems, except Otter, ran in the TFA demonstration division of CASC-J5 [5]. Their system descriptions can be found on the CASC-J5 web site. Testing was done on a computer with a 2.8GHz Intel Xeon CPU, 3GB memory, running Linux FC8, with a 300s CPU time limit. Table 2 summarizes the results. The table shows 60 theorems are proved by all systems, and 100 by none; none of the systems can establish non-theoremhood; SPASS-XDB—0.8 gets the most timeouts, i.e., the most problems that might be solved with a larger time limit; with the exception of leanCoP-Omega, average times for solutions are very low; and SPASS+T is the system to use if you want a quick answer. Examination of the individual problem results shows that SPASS-XDB's strengths (relative to the other systems) are on systems of equations and inequations (thanks to Mathematica), and high compliance to the TPTP arithmetic standard. SPASS-XDB's major weaknesses are proving

System	THM	CSA	TMO	UNK	Avg CPU for solns	Unique solns
Problems	810	65				
SPASS-XDB—0.8	558	0	218	99	1.22	15
SPASS+T—2.2.12	499	0	0	376	0.00	55
SNARK—20080805r027	482	0	143	250	0.34	5
MetiTarski—1.7	372	0	2	501	0.57	6
leanCoP-Omega—0.1	304	0	163	408	19.35	1
Otter—3.3	91	0	69	715	0.97	0
Union	775	0	267	814		82
Intersection	60	0	0	10		

Table 2. TFA Results

universal claims, and dealing with arithmetic mixed with uninterpreted symbols. SPASS+T and MetiTarski have strengths that complement SPASS-XDB's weaknesses, which contributes to the larger union of theorems proved. It is conjectured that SPASS+T's use of an SMT solver for arithmetic with uninterpreted symbols is a source of its strong performance.

5 Conclusion

SPASS-XDB's new arithmetic reasoning capabilities will be useful in SPASS-XDB primary mission: applications of reasoning with world knowledge. Furture work includes improving mathematical reasoning mixed with uninterpreted symbols, and solving universal mathematical conjectures. Additionally, SPASS-XDB is currently being extended to obtain world knowledge axioms from web search.

References

1. M. Beeson. The Mechanization of Mathematics. In C. Teuscher, editor, *Alan Turing: Life and Legacy of a Great Thinker*, pages 77–134. Springer-Verlag, 2004.
2. D. Gunning, V. Chaudhri, and C. Welty. Introduction to the Special Issue on Question Answering. *AI Magazine*, 31(3):11–12, 2010.
3. M. Suda, G. Sutcliffe, P. Wischnewski, M. Lamotte-Schubert, and G. de Melo. External Sources of Axioms in Automated Theorem Proving. In B. Mertsching, editor, *Proceedings of the 32nd Annual Conference on Artificial Intelligence*, number 5803 in Lecture Notes in Artificial Intelligence, pages 281–288, 2009.
4. G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure. The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.
5. G. Sutcliffe. The 5th IJCAR Automated Theorem Proving System Competition - CASC-J5. *AI Communications*, page To appear, 2011.
6. C. Weidenbach, A. Fietzke, R. Kumar, M. Suda, P. Wischnewski, and D. Dimova. SPASS Version 3.5. In R. Schmidt, editor, *Proceedings of the 22nd International Conference on Automated Deduction*, number 5663 in Lecture Notes in Artificial Intelligence, pages 140–145. Springer-Verlag, 2009.