Enabling Symbolic and Numerical Computations in HOL Light

Ons Seddiki(✉), Cvetan Dunchev, Sanaz Khan-Afshar, and Sofiène Tahar

Department of Electrical and Computer Engineering, Concordia University,
1455 de Maisonneuve W., Montreal, QC H3G 1M8, Canada
{o_sed,dunchev,s_khanaf,tahar}@ece.concordia.ca

Abstract. Verifying mathematical statements by interactive theorem provers often requires algebraic computation. Since many Mechanized Mathematical Systems (MMS) support the OpenMath standard, we propose to link the HOL Light theorem prover to other MMSs via OpenMath. In particular, we present an interface between HOL Light and Mathematica enabling HOL Light users to evaluate arithmetic, transcendental and linear algebraic expressions, using Mathematica.

1 Introduction

Theorem proving is a technique which proves or checks the validity of logical statements. It is based on sequential applications of sound inference rules to a given axiomatic system. The statements proved by the theorem prover, accepting that its core is sound, are absolutely accurate in contrast to paper-and-pencil methods or computer simulations. Often in the process of interactive theorem proving one needs to perform a symbolic computation which might be a tedious task requiring hundreds of inference rules. For example, computing the value of a polynomial over $\mathbb{R}$ needs many inference rules and auxiliary theorems over the theory of real numbers. Furthermore, computing the roots of the same polynomial by the theorem prover is a very hard task. To avoid such limitations, one may make use of a Computer Algebra System (CAS) which has the needed functionality to perform the computation. The result of the CAS is transformed to an axiom which is added to the list of axioms and used by the theorem prover.

Many researchers have addressed the issue of combining symbolic/numeric computation with logical reasoning. One solution is building a CAS inside a theorem prover (e.g., [7]) or building a theorem prover inside a CAS (e.g., [2,10]). The second approach implements a bridge between theorem provers and CAS (e.g., PVS and Maple [1], Isabelle and Maple [3], and HOL and Maple [6]). This connection involves a master-slave relation in which the theorem prover is usually considered as a master and the CAS as a slave, with the assumption that there is no trust in the CAS. The third approach (e.g. Mathscheme[1]) is to build an integrated framework that provides the functionalities of both CAS and theorem proving integrating them into a single tool without sacrificing the soundness

[1] http://www.cas.mcmaster.ca/research/mathscheme/.

M. Kerber et al. (Eds.): CICM 2015, LNAI 9150, pp. 353–358, 2015.
DOI: 10.1007/978-3-319-20615-8_27

and without using an intermediate language. Finally, the fourth approach is to define a framework using a standard for mathematical information (such as MathML[2] and OpenMath[3]) that can be exchanged between different Mechanized Mathematical Systems (MMS). For instance, in [4] the authors used OpenMath to develop a Java client-server applet between Maple as a client and the Lego theorem prover as a server.

In this paper, we propose a tool linking HOL Light[4] to Mathematica[5] through the OpenMath standard. In contrast to [4] where the authors present a Java applet which takes a Maple expression as input and returns a Lego expression through the translation to OpenMath, our work is a combination between two external tools where OpenMath is used as a middleware. Another difference between our approach and [4] is that we do not rely on the communication layer established between the client and the server, but on the direct translation of a HOL Light statement to a Mathematica term and vice-versa. Therefore, the performance of the computation is increased.

The proposed tool is part of a general framework providing a heterogeneous problem-solving environment, which connects HOL Light to any MMS. Figure 1 illustrates the general approach of this framework which encompasses a variety of MMSs that support OpenMath such as the theorem provers LEGO[6] and COQ[7], the CASs Maple[8], Gap[9] and Mathematica, or the numerical solver Mupad[10] with the intention of solving and reasoning over larger sets of problems.

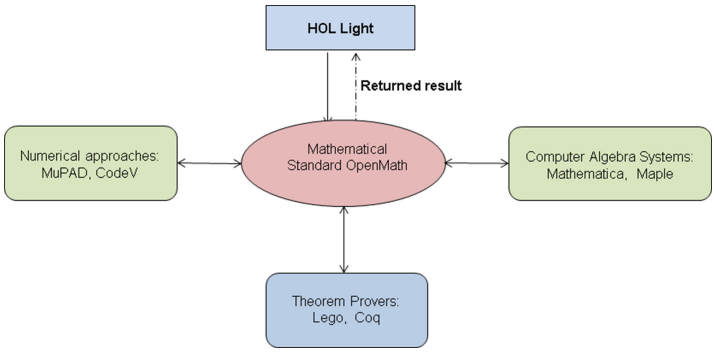

Fig. 1. Connecting Different MMS to HOL Light using OpenMath

[2] http://www.w3.org/Math/.
[3] http://www.openmath.org/overview/index.html.
[4] http://www.cl.cam.ac.uk/~jrh13/hol-light/.
[5] http://www.wolfram.com/.
[6] http://www.dcs.ed.ac.uk/home/lego/.
[7] https://coq.inria.fr/.
[8] http://www.maplesoft.com/products/maple/.
[9] http://www.gap-system.org/.
[10] http://de.mathworks.com/discovery/mupad.html.

2 Tool Description

The proposed linkage tool starts by translating the HOL Light statement into an OpenMath object. Then, a Java phrasebook [5], which is a collection of encoding/decoding methods between OpenMath and Mathematica, converts the OpenMath object into an expression, that is passed to Mathematica. The computation of Mathematica is translated back to an OpenMath object using again the Java phrasebook. Finally, the latter is parsed by our tool and converted to a HOL Light axiom. We developed a translator from HOL Light to OpenMath and visa-versa, which enables HOL Light users to access Mathematica's kernel using the Phrasebook OpenMath-Mathematica proposed by Caprotti [5]. After the computation, the returned result from Mathematica is represented as an axiom in HOL Light tagged by `Mathematica` in the form `Mathematica` $\vdash \Psi$, where Ψ is the expression performed by Mathematica. Moreover, each theorem derived from this axiom inherits the tag `Mathematica`. This procedure helps to easily trace the axioms created from the interaction with an external tool. After the computation, the returned result can also be represented in another form as a sub-goal and added to the assumption of a main goal. One needs to prove it in order to pursue further proofs. Figure 2 depicts the structure of the tool connecting HOL Light to Mathematica, which is comprised of the following three modules:

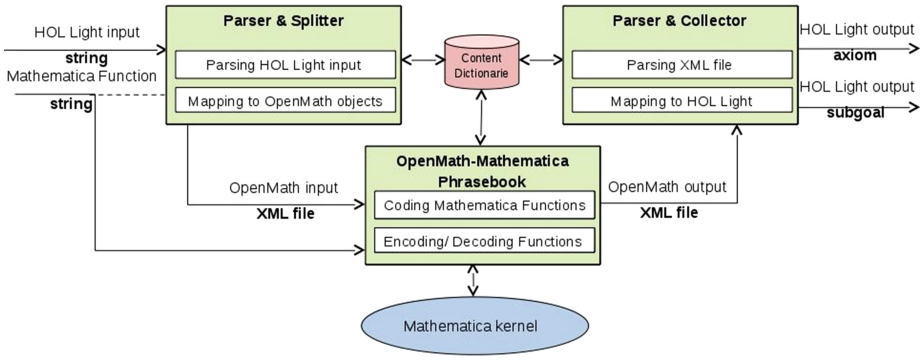

Fig. 2. Tool structure

The *Parser & Splitter* transforms the HOL Light statement into a corresponding OpenMath object as understood by means of the Content Dictionaries (CDs)[11]. First, it parses the HOL Light expression according to a grammar [9] which converts a HOL Light expression to the corresponding OpenMath object. Then, it decomposes the HOL Light input statement into a list of operations and operands. Thereafter, it maps each element of the list with the corresponding OpenMath symbol as understood by means of the related CDs. Finally, it stores the description of the OpenMath object in an XML file.

[11] http://www.openmath.org/cd/.

The *OpenMath-Mathematica Phrasebook*[12] defines a collection of Java classes, which provide two sets of methods. The first set represents the encoding and decoding methods between OpenMath and Mathematica based on the declaration of the corresponding CDs. The second one describes the built-in Mathematica call function with the tag already specified by the user. The phrasebook translates the XML file that describes the OpenMath input object into a Mathematica statement, which is then passed to the Mathematica kernel through a Mathematica service. (See footnote 12) This service allows users to remotely call the Mathematica kernel as a computational engine using MathLink.[13] This connection is established via the TCP/IP protocol. Once the result is computed, the Mathematica output statement is translated back to OpenMath and an XML file is generated.

The *Parser & Collector* translates the OpenMath object which encodes the output of Mathematica into the corresponding HOL Light symbols in the relevant CDs. Then, it collects all the HOL Light symbols and returns an axiom tagged by the name of the CAS (i.e., `Mathematica`). In other cases, we can generate the returned result as a sub-goal and prove it in HOL Light. This provides some kind of a determinism to the proof process, because when one knows the result of the computation, finding the proof is more straightforward rather than searching for the result during the whole process of proof derivation.

The above process is sound in the sense that it preserves the types during the parsing and passing of the data. The HOL Light statements are by definition well typed. For example, the HOL Light expression "`x pow 3 + (&2 * x) pow 2 + x`" represents the polynomial "$x^3+(2x)^2+x$" over the field of the reals. Based on this fact, the Parser & Splitter module converts the HOL Light expression into the corresponding OpenMath object preserving the types. The Java Phrasebook also preserves the types. Finally, the OpenMath object obtained from Mathematica is converted to a HOL Light axiom and all types are preserved because we know in advance the types of all supported functions.

3 Applications

We have used our tool on several examples like solving or evaluating non-closed form formulas such as arithmetic or polynomial manipulations or matrix operations, simplifying integrals, derivatives and transcendental functions, computing eigenvectors, checking inequalities, finding roots and factorization of complex polynomials. In the following, we give two simple examples. The input is a string which represents an expression given to HOL Light. The expression consists of two parts. The first part is an ordinary HOL Light expression, whereas the second part represents the built-in Mathematica symbol such as "Factor", "Simplify", etc. The output is an axiom in HOL Light. For example, using the built-in symbol "FullSimplify" we show the computation of the integral $\int_1^{10}(x+1)\,\mathrm{d}x$:

[12] http://mathdox.org/new-web/index.html.

[13] http://reference.wolfram.com/mathematica/tutorial/IntroductionToMathLink.html.

Input:
`#call_mathematica` "`real_integral (real_interval` [&1, &10]) ($\backslash x.x + \&1$)" "`FullSimplify`";;
The computed result, 117/2, is returned to HOL Light as an axiom:
Output:
`val it: thm = Mathematica` ⊢ `real_integral (real_interval` [&1, &10]) ($\backslash x.x$ + &1) = &117/&2
The next example shows the factorization of the polynomial $x^3 + 2.x^2 + x$:
Input:
`#call_mathematica` "x *pow* 3 + &2 ∗ (x *pow* 2) + x" "`Factor`";;
The expected result, $x.(x + 1)^2$, is returned as a HOL Light axiom:
Output:
`val it: thm = Mathematica` ⊢ x *pow* 3+&2∗(x *pow* 2) +x = x∗(&1+x) *pow* 2

We have conducted several more comprehensive experiments, which can be found in [9]. Moreover, our tool was successfully applied in the formal verification of optical systems [8], where we send from HOL Light the expression of a boundary condition of an optical interface described with electromagnetic fields to Mathematicia in order to be simplified. Details of these experiments can be found in [9]. These examples emphasize not only the benefits of computing such Mathematica expressions within HOL Light but also the efficient performance of our tool in terms of execution time. Our tool, called *HolMatica*, is implemented in a way that we can easily adapt it to any other CAS or theorem provers that support OpenMath. The *HolMatica* tool and running examples can be downloaded from http://hvg.ece.concordia.ca/research/tools/holmatica/

References

1. Adams, A., Dunstan, M.N., Gottliebsen, H., Kelsey, T., Martin, U., Owre, S.: Computer algebra meets automated theorem proving: integrating maple and PVS. In: Boulton, R.J., Jackson, P.B. (eds.) TPHOLs 2001. LNCS, vol. 2152, pp. 27–42. Springer, Heidelberg (2001)
2. Bauer, A., et al.: Analytica - an experiment in combining theorem proving and symbolic computation. JAR **21**(3), 295–325 (1998)
3. Ballarin, C., et al.: Theorems and Algorithms: An Interface between Isabelle and Maple. In: ISSAC, pp. 150–157. ACM (1995)
4. Caprotti, O., Cohen, A.M.: Integrating computational and deduction systems using OpenMath. ENTCS **23**(3), 469–480 (1999)
5. Caprotti, O., Cohen, A.M., Riem, M.: Java phrasebooks for computer algebra and automated deduction. SIGSAM Bulltin **34**, 33–37 (2000)
6. Harrison, J., Théry, L.: A skeptic's approach to combining HOL and maple. JAR **21**, 279–294 (1998)
7. Kaliszyk, C., Wiedijk, F.: Certified computer algebra on top of an interactive theorem prover. In: Kauers, M., Kerber, M., Miner, R., Windsteiger, W. (eds.) MKM/CALCULEMUS 2007. LNCS (LNAI), vol. 4573, pp. 94–105. Springer, Heidelberg (2007)
8. Afshar, S.K., et al.: formal analysis of optical systems. MCS **8**(1), 39–70 (2014)

9. Seddiki, O.: Linking HOL Light to Mathematica using OpenMath. Master's thesis, Concordia University, Montreal, QC, Canada, October 2014
10. Windsteiger, W.: Theorema 2.0: a graphical user interface for a mathematical assistant system. CEUR Workshop Proceedings, vol. 118, pp. 73–81 (2012)