Extending the HOL theorem prover with a Computer Algebra System to Reason about the Reals

John Harrison* & Laurent Théry†

University of Cambridge Computer Laboratory
New Museums Site
Pembroke Street
Cambridge CB2 3QG
England.

Phone: (+44) 223 334760

Email: {jrh,tl}@cl.cam.ac.uk

Abstract

In this paper we describe an environment for reasoning about the reals which combines the rigour of a theorem prover with the power of a computer algebra system.

1 Introduction

Computer theorem provers are a topic of research interest in their own right. However much of their popularity stems from their application in computer-aided verification, i.e. proving that designs of electronic or computer systems, programs, protocols and crypto-systems satisfy certain properties.

Such proofs, as compared with the proofs one finds in mathematics books, usually involve less sophisticated central ideas, but contain far more technical

*Supported by the Science and Engineering Research Council, UK.
†Supported by SERC grant GR/G 33837 and a grant from DSTO Australia.

details and therefore tend to be much more difficult for humans to write or check without making mistakes. Hence it is appealing to let computers help.

Some fundamental mathematical theories, such as arithmetic, are usually required for such proofs, and in particular the real numbers are useful in several fields:

- The verification of floating-point hardware. It is often desirable to specify the intended behaviour of a floating-point unit with reference to the real numbers which the bit-strings approximate.
- The verification of systems incorporating linear components, such as vehicle braking mechanisms or nuclear reactor controllers.
- Extension of theorem-proving methods into the realm of day-to-day applied mathematics as undertaken by engineers and scientists.

It is mainly the last application which will concern us here. Computer Algebra Systems (CASs) are very popular for such applications as multi-precision arithmetic, differentiation and integration, and the expansion of functions in power series. However they have two drawbacks.

Firstly, they tend to have no concept of a logical language in which to state precise mathematical theorems. Usually, they accept an algebraic expression and return another purportedly equal to it. They may not even mean equality in the true sense; there may be restrictions, or 'equality' may mean some weaker notion such as equality as meromorphic functions, e.g. $(x^2-1)/(x-1) = x+1$. By contrast, theorem provers are fundamentally based on logic, and offer good facilities for the management and organization of large proofs.

Secondly, even where the result from a CAS has some precise and unambiguous meaning, it may be false! CASs tend to simplify expressions fairly aggressively, taking shortcuts which are not always rigorously justified. In any case, they implement some extremely complicated algorithms to get their results, and it would be surprising if there were no bugs. For example Maple [6] evaluates $\int_{-1}^{1} \sqrt{x^2}\ dx$ to 0. Mathematica [21] gets this integral right and returns 1, but it returns 0 when given $\int_{-1}^{1} \frac{1}{\sqrt{x^2}}\ dx$, forgetting the singularity at 0. Computer theorem provers can offer much greater reliability; this is particularly so with strictly foundational systems like HOL.

However there is one obvious advantage that CASs have over theorem provers: they have many powerful decision procedures and heuristics which make them effective to use. To emulate these in a computer theorem prover would be an enormous task, and the burden of producing a logical proof would doubtless slow down the algorithms a great deal.

In this paper, we present an approach that combines the best of the two worlds by a careful exploitation of a link between HOL and a CAS. In the next section we outline the theoretical background to the formal development of integral calculus in HOL. We then consider the technical aspects of the link and how it might be useful. In the final section we draw the threads together by showing how we can evaluate certain trigonometric integrals in an entirely rigorous manner.

2 The reals in HOL

HOL is strictly definitional: rather than the existence of a new mathematical structure being asserted, the structure is *defined* in terms of existing structures, and the characterizing 'axioms' are derived by formal proof. For example, the real numbers have been constructed in HOL using a version of Dedekind's construction; a brief overview is given in [10], which sketches the procedure and discusses some significant parts of elementary analysis, including differentiation. Here we concentrate on the extension of this work to include integration, which we will use in a later example.

A consequence of the definitional approach is that we must be particularly careful about the way we define mathematical notions. In some cases, the appropriate definitions are uncontroversial. However many areas of mathematics offer a range of subtly different approaches. Integration is a particularly difficult case (its history is traced in [7] and [19]). Many people think of integration as the opposite of differentiation. Undergraduate mathematics courses usually present the Riemann integral. At a more advanced level, Lebesgue theory seems dominant; consider the following quote from [4]

> It has long been clear that anyone who uses the integral calculus in the course of his work, whether it be in pure or applied mathematics, should normally interpret integration in the Lebesgue sense. A few simple principles then govern the manipulation of expressions containing integrals.

We shall consider these notions in turn and explain our selection of the Kurzweil-Henstock gauge integral. For our purposes it is particularly important to get clear the relationship between differentiation and integration. Ideally we would like the Fundamental Theorem of Calculus

$$\int_a^b f'(x)dx = f(b) - f(a)$$

to be true whenever f is differentiable at all points of the interval $[a, b]$ and its derivative at x is $f'(x)$.

The Newton Integral

Newton actually *defined* integration as the reverse of differentiation. Integrating f means finding a function which when differentiated, gives f (called an *antiderivative*). Therefore the Fundamental Theorem is trivially true for the Newton Integral.

Newton's approach however has certain obvious defects as a formalization of the notion of the area under a curve. It is not too hard to prove that all derivatives are *Darboux continuous*, that is, they attain all intermediate values. Consequently, a simple step function:

$$f(x) = \begin{cases} 0 & \text{if } x < 1 \\ 1 & \text{if } x \geq 1 \end{cases}$$

which intuitively has a perfectly well-defined area, does not have a Newton integral.

The Riemann Integral

The Riemann integral defines the area under a curve in terms of the areas of strips bounded by the curve, as the width of the strips tends to zero. It handles the step function but has other defects. Infinite limits have to be written as limiting cases of other integrals in various ad-hoc ways. The integral does not have convenient convergence properties: limits of sequences of integrable functions can fail to be integrable. And, particularly relevant to the present work, the Fundamental Theorem of Calculus fails to hold (the example given below for the Lebesgue integral also serves for the Riemann).

The Lebesgue Integral

The Lebesgue integral is superior to the Riemann integral in a number of important respects. It accommodates infinite limits without any ad-hoc devices, and obeys some useful convergence theorems. Any (directly) Riemann integrable function is also Lebesgue integrable, and some functions which have no Riemann integral nonetheless have a Lebesgue integral, the classic example being the indicator function of the rationals:

$$f(x) = \begin{cases} 1 & \text{if } x \in \mathbb{Q} \\ 0 & \text{if } x \notin \mathbb{Q} \end{cases}$$

One feature that the Lebesgue integral shares with the Riemann integral is that the Fundamental Theorem of Calculus is *still* not generally true. The following counterexample was given in Lebesgue's thesis:

$$f(x) = \begin{cases} 0 & \text{if } x = 0 \\ x^2 sin(1/x^2) & \text{if } x \neq 0 \end{cases}$$

This is an inevitable consequence of the fact that the Lebesgue integral, in common with the Riemann integral, is an *absolute* integral, meaning that whenever f is integrable, so is $|f|$.

Other Integrals

Various integrals have been proposed which extend the Lebesgue integral and for which the Fundamental Theorem is true. The first was Denjoy integral [15] which, starting with the Lebesgue integral, constructs a sequence of integrals by a process of transfinite recursion which Denjoy called '*totalisation*'. A very simple characterization of the Denjoy integral was given by Perron [3], but it is not constructive and the development of the theory uses theorems about the Lebesgue integral.

The Kurzweil-Henstock Gauge Integral

Surprisingly recently it was observed that a simple modification of the Riemann limit process could give an integral equivalent to the Denjoy and Perron integrals. This seems to have first been made explicit by Kurzweil [17], but its later development, in particular the proof of Lebesgue-type convergence theorems, was mainly due to Henstock [11]. It is known as the 'Kurzweil-Henstock gauge integral' or simply 'gauge integral'. In the following, we give a sketch of the definition of this integral following the terminology given in [18] and note some results that have already been proved in HOL. An fuller introduction may be found in the undergraduate textbook [8] or the definitive [12].

The limiting process involved in the gauge integral seems rather obscure at first sight, but the intuition can be seen quite clearly if we consider integrating a derivative. Suppose f is differentiable for all x lying between a and b. Then given any such x and any $\epsilon > 0$, we know that there exists a $\delta > 0$ such that whenever $|y - x| < \delta$

$$\left| \frac{f(y) - f(x)}{y - x} - f'(x) \right| < \epsilon$$

For some fixed ϵ, this δ can be considered as a function of x which always returns a strictly positive real number, i.e. a *gauge*.

Consider now splitting the interval $[a, b]$ into a *tagged division*, i.e. a finite sequence of non-overlapping intervals, each interval $[x_i, x_{i+1}]$ containing some nominated point t_i called its *tag*. We shall say that a division is δ-fine (or fine with respect to a gauge δ) if for each interval in the division:

$$[x_i, x_{i+1}] \subseteq (t_i - \delta(t_i), t_i + \delta(t_i))$$

It is an important property that a δ-fine division exists for any gauge. This has been proved in HOL using Bolzano's technique of bisection, as detailed in [10].

Now we say that f has *gauge integral* I on the interval $[a, b]$ if for any $\epsilon > 0$, there is a gauge δ such that for any δ-fine division, the usual Riemann-type sum approaches I closer than ϵ:

$$\left|\sum_{i=0}^{n} f(t_i)(x_{i+1} - x_i) - I\right| < \epsilon$$

The property that any derivative has a gauge integral has been proved in HOL starting from the basic definition. It has also been proved that the Fundamental Theorem holds.

As hinted earlier, the gauge integral is nonabsolute, but it has all the attractive convergence properties of the Lebesgue integral. There is quite a simple relationship between the two integrals: f has a Lebesgue integral precisely when both f and $|f|$ have a gauge integral. A more surprising connection is that modifying the above definition to insist that the tag is one endpoint of the interval gives exactly the Lebesgue integral, but without requiring any of the usual measure-theoretic machinery.

3 How to use a CAS inside a theorem prover?

How we use a CAS inside a theorem prover depends on the degree of confidence about the accuracy of the answers given by the CAS. Roughly there are three possible attitudes: to trust the CAS completely, to trust it partially, or to not trust it at all.

Complete trust

Trusting the CAS completely means that given a request A, if the CAS's answer is B, then the theorem prover has to accept the theorem $\vdash A = B$. Implementing this solution in HOL violates the usual strict principle of deriving all theorems from the axioms. However a technique given in [9] can be applied to get around this problem. This technique consists in defining a constant CAS logically equivalent to false. With this constant, it is then possible to produce theorems of the form $CAS \vdash A = B$ which are (trivially) true. The first advantage of using this technique is that the generated theorems have a natural reading: "providing the CAS is correct, A equals B". The second advantage is that every theorem that uses such a theorem in its proof will automatically inherit the assumption CAS. This means that there is a simple criterion to differentiate 'purely true' theorems from theorems proved with the help of the CAS.

Partial trust

The notion of partial trust comes from efficiency considerations. Algorithms in CASs can be described as optimized algebraic manipulations on ad-hoc data structures. Trying to implement the same algorithms in a prover where the notions of genericity and consistency are preponderant usually entails an important degradation of performance. This degradation may jeopardize the interactive use of the theorem prover where a reasonable reaction time is mandatory. A typical example of such a situation can be found in arithmetic with HOL's REDUCE library. This library provides an automatic procedure to compute ground arithmetic terms. Given a term, it returns a theorem establishing the equality between this term and its reduced form. In order to produce such a theorem, it uses the primitive inference rules of Peano arithmetic. This means that on practical examples, the procedure tends to be rather slow. For example it takes more than 10 seconds to obtain the theorem $\vdash 2^{10}/2^9 = 2$ from the term $2^{10}/2^9$. By contrast, most of the CASs return this result in less than 0.1 second.

The technique of partially trusting the CAS consists in accepting the result of the CAS during the interactive proof of the proposition, but finally when the theorem has to be recorded, *checking* all the results that have been computed by the CAS. The benefit of such a technique is to combine an efficient interaction with the theorem prover with the security of the result. The implementation of such a technique is a direct application of the lazy approach developed by Richard Boulton [2]. This approach gives a natural framework for delaying the proofs of some subgoals. The justification of these subgoals can then be postponed and handled by a batch process where the efficiency requirements are less stringent.

No trust at all

The main drawback of the previous approach is that as noted above, the CAS may sometimes generate wrong answers. The benefit of speeding up the interaction using the CAS can then be overwhelmed by the time spent on trying to prove unprovable subgoals. The only way to avoid this kind of situation is never to trust the CAS's answers. Even in this case, having a CAS can be very useful, since there are many situations where finding a solution of a problem is computationally more complex than verifying it. Then we can delegate the 'finding' part of the procedure to the CAS and automate the 'checking' part in the prover with an efficient procedure. The CAS is then used as an oracle; one might regard it as a proof planner, though in general these guide the detailed structure of the proof, rather than just selecting the starting point.

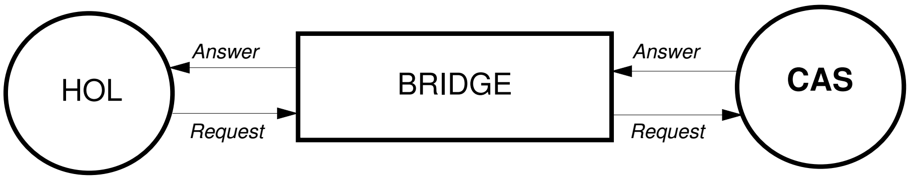
Conclusion

The first solution is convenient to give a simple integration between HOL and a CAS without actually generating false theorems. Proofs using the CAS are marked in a clear and simple way. However it is not secure because a determined user may create any theorem with the assumption CAS. This sort of pragmatic approach has already been used to delegate some low-level parts of hardware proofs to other tools [20]. The second is primarily useful for simple arithmetic, where a wrong answer is almost inconceivable and the insistence on carrying the proof through in the theorem prover at all would be questioned by many. The third alternative is the one we are most interested in here. It has the merit of only using the CAS as a guide; the proof itself is still done rigorously in the usual manner. In real analysis, these situations occur quite frequently for problems like integration or factorization. The example given at this end of the paper illustrates this idea.

4 Implementation Issues

Connecting a theorem prover with other systems is not a new idea. Some experiments have already been done in linking HOL with other theorem provers (see [1] and [16] for example). Our application differs slightly from these previous experiments. First of all, the relation between HOL and the CAS is a clear master/slave relation: the use of the CAS is limited to some precise tasks (mainly algebraic simplifications) within an overall proof. Another difference is that the requests sent to the CAS only concern standard algebraic expressions. The translation from the HOL term representation to the CAS one is then straightforward and involves mostly minor syntactic modifications.

The actual implementation of the link between HOL and the CAS follows the lines given in [5] and can be depicted by the following drawing:



The organization involves three different processes: HOL, the CAS and a bridge. The communication between them uses broadcasting messages. HOL and the CAS send and receive messages with string representing formulas in their own syntax. For example, $(\sin x + \cos x)'$ is represented as `diff(sin(x)+cos(x),x)` in Maple, as `D[Sin[x]+Cos[x],x]` in Mathematica but as `deriv x.(sin x)+(cos x)` in HOL. The role of the bridge is to perform the data coercion automatically. Communicating by broadcasting makes the prover completely independent of the CAS: HOL just requests an answer from an oracle. This model allows us not only to experiment with connecting different CASs without any perturbation on the prover side, and furthermore it actually transforms the CAS into a server. Thus it is possible to share this server between several HOL sessions by connecting these sessions to the same CAS.

From the prover's point of view, the access to the CAS is represented by a single function `call_CAS` whose type is `term -> string -> term`. Its first argument is the term to transform, the second argument represents the method to apply. In the current implementation, only two methods are available. `SIMPLIFY` gives the answer as if the term had been typed at the top level of the CAS. `FACTORIZE` tries to factorize the term. With the function `call_CAS`, it is then possible to compute some arithmetic expressions via the CAS:

```
# call_CAS "(((FACT 5) EXP 2) - 1) MOD (3 EXP 2)" `SIMPLIFY`;;
"8" : term
Run time: 0.1s

# call_CAS "(x*x)+(7*x)+12" `FACTORIZE`;;
"(x + 3) * (x + 4)" : term
Run time: 0.1s
```

5 An example

In order to illustrate how the combined system can be used, we explain the construction of a *secure* procedure to automatically integrate polynomial expressions in cosine and sine. This procedure is composed of five different steps. The steps 2, 4 and 5 are proof steps, i.e. steps that produce a theorem. The steps 1 and 3 are CAS steps, i.e. steps that produce only a result. This procedure is secure in the sense that as described in section 3 it never trusts the CAS. In order to have a concrete example to explain what every step is supposed to do, we are going to apply this procedure to compute the following integral:

$$\int_0^t \sin^3 u \; du$$

The first step of the procedure is to call the CAS to get the actual value of this integral. With our example, we get the result:

$$\int_0^t \sin^3 u \; du = -\frac{1}{3}\sin^2 t \cos t - \frac{2}{3}\cos t + \frac{2}{3}$$

If we can *prove* that the derivative of the right part of this equality is in fact $\sin^3 x$, we get this result as a theorem by applying the Fundamental Theorem of Calculus (see section 2). Step 2 consists in computing this derivative inside the prover. This is done with an automated conversion which composes the standard theorems about the differentiation of sums, products, cosine and sine. Step 2 on our example gives the theorem:

$$\vdash (-\frac{1}{3}\sin^2 t \cos t - \frac{2}{3}\cos t + \frac{2}{3})' = -\frac{1}{3}\,(2\sin t \cos t \cos t - \sin^3 t) + \frac{2}{3}\sin t$$

In order to prove that this derivative is in fact $\sin^3 t$, the remaining steps have to establish the following theorem:

$$\vdash -\frac{1}{3}\,(2\sin t \cos t \cos t - \sin^3 t) + \frac{2}{3}\sin t - \sin^3 t = 0$$

In order to do this, we transform this expression in a polynomial by replacing $\sin t$ by x and $\cos t$ by y, and then factorize it with the help of the CAS. This is step 3 and we get the result:

$$-\frac{1}{3}\,(2\,x\,y\,y - x^3) + \frac{2}{3}\,x - x^3 = -\frac{2}{3}\,x\,(y^2 + x^2 - 1)$$

In step 4, this result is proved automatically inside the prover. The procedure expands out products and collects together monomials of the same degree. The coefficients should then all be zero; this is checked by a simple extension of the REDUCE library to rational arithmetic. Applied to the example, we get:

$$\vdash -\frac{1}{3}\,(2\,x\,y\,y - x^3) + \frac{2}{3}\,x - x^3 = -\frac{2}{3}\,x\,(y^2 + x^2 - 1)$$

Finally as x is $\sin t$ and y is $\cos t$ we have $y^2 + x^2 - 1 = 0$. Furthermore, such a factor will occur in any trigonometric identity of this form, so we expect the procedure to work. From this, we deduce in step 5 the following theorem:

$$\vdash -\frac{1}{3}\,(2\sin t\cos t\cos t - \sin^3 t) + \frac{2}{3}\sin t = \sin^3 t$$

By composition of this theorem, the theorem of step 2 and the Fundamental Theorem of Calculus, we get the expected theorem:

$$\vdash \int_0^t \sin^3 u\ du = -\frac{1}{3}\sin^2 t\cos t - \frac{2}{3}\cos t + \frac{2}{3}$$

Although this theorem has been completely derived from the definitions, the use of the CAS has allowed us to obtain it from a simple set of tools, namely a procedure to compute derivatives and a decision procedure about the equality of two polynomials. This procedure can then be applied to compute any polynomial integral. Figure 1 gives some results on different examples in term of run-time and number of primitive inferences used to integrate them.

6 Conclusions and Future Work

The example of section 5 shows the immediate benefit of having a CAS inside a prover. Some operations like integration and factorization become available inside the prover without jeopardizing the security of the result. The performance, as can be seen in the last integrations of figure 1, is still below acceptable standard. Performing differentiation and collecting monomials are a small part of this; what dominates is doing rational arithmetic. It is our intention to implement a binary representation inside the HOL logic, which would speed up arithmetic on large numbers enormously without sacrificing security. It is also our intention to tackle

Value	Run time	Intermediate theorems generated
$\int_0^t \sin u \; du$	6.3s	1901
$\int_0^t \sin u \cos u \; du$	7.8s	2276
$\int_0^t \sin^2 u \; du$	20.4s	6442
$\int_0^t \sin^3 u \; du$	24.6s	7707
$\int_0^t \sin^4 u \; du$	49.4s	16527
$\int_0^t \sin^5 u \; du$	70.4s	25387
$\int_0^t \sin^6 u \; du$	110.4s	40814
$\int_0^t \sin^7 u \; du$	301.4s	125093
$\int_0^t \sin^8 u \; du$	2345.2s	1019832

Figure 1: Integration Benchmark for Sun4/ Allegro Common Lisp

more realistic examples where the proof management of the theorem prover and the heuristics of the CAS can be fully exploited.

From the implementation's point of view, the link between HOL and a CAS has mainly been with the Maple system [6]. But it already seems probable that having only one CAS is not sufficient as there is not one best and universal CAS but rather different systems which specialize in different areas. In order to make the best features of these different CASs available inside HOL, a possible extension is to handle multiple connection via an auction mechanism as in [14]. In that system, all the requests are sent simultaneously to different CASs and then the auction mechanism is applied to select the best of all the answers. This extension and an alternative connection with the AXIOM system [13] are under examination.

References

[1] M. Archer, G. Fink, L. Yang, *Linking Other Theorem Provers to HOL Using PM: Proof Manager*, Proceedings of the 1992 International Workshop on the HOL Theorem Proving System and its Applications, North-Holland 1993.

[2] R. J. Boulton, *A Lazy Approach to Fully-Expansive Theorem Proving*, Proceedings of the 1992 International Workshop on the HOL Theorem Proving System and its Applications, North-Holland 1993.

[3] P. Bullen, *Non-absolute integrals: A survey*, Real Analysis Exchange, vol. 5 pp. 195-259, 1980.

[4] J. C. Burkill, *The Lebesgue Integral*, Cambridge Tracts in Mathematics and Mathematical Physics no. 44, Cambridge University Press 1965.

[5] D. Clément, F. Montagnac, V. Prunet, *Integrated Software Components: a Paradigm for Control Integration*, Proceedings of the European Symposium on Software Development Environments and CASE Technology, Königwinter, Springer-Verlag LNCS 509, June 1991.

[6] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan, S. M. Watt, *A Tutorial Introduction to Maple V*, Springer-Verlag, 1992.

[7] D. van Dalen, A. F. Monna, *Sets and Integration: an Outline of the Development*, Wolters-Noordhoff 1972.

[8] J. DuPree, C. Swartz, *Introduction to Real Analysis*, Wiley 1988.

[9] M. Gordon, Private Communication.

[10] J. R. Harrison, *Constructing the Real Numbers in HOL*, Proceedings of the 1992 International Workshop on the HOL Theorem Proving System and its Applications, North-Holland 1993.

[11] R. Henstock, *A Riemann-type integral of Lebesgue power*, Canadian Journal of Mathematics vol. 20 pp. 79-87, 1968.

[12] R. Henstock, *The General Theory of Integration*, Clarendon Press, Oxford 1991.

[13] R. D. Jenks, R. S. Sutor, *AXIOM: the Scientific Computation System*, Springer-Verlag 1992.

[14] N. Kajler, *Cas/Pi: A Portable and Extensible Interface for Computer Algebra Systems*, Proceedings of ISSAC '92, ACM 1992.

[15] A. Kechris, *The Complexity of Antidifferentiation, Denjoy Totalization, and Hyperarithmetic Reals*, Proceedings of the International Conference of Mathematicians, Berkeley 1986.

[16] R. Kumar, T. Kropf, K. Schneider, *Integrating a First-Order Automatic Prover in the HOL environment*, Proceeding of the 1991 International Workshop on the HOL Theorem Prover System and its Applications, IEEE Computer Society Press 1991.

[17] J. Kurzweil, *Generalized Ordinary Differential Equations and Continuous Dependence on a Parameter*, Czechoslovak Mathematics Journal vol. 7(82), pp. 418-446, 1958.

[18] E. J. McShane, *A Unified Theory of Integration*, American Mathematical Monthly 80 pp. 349-357, 1973.

[19] I. N. Pesin, *Classical and modern integration theories*, Academic Press 1970.

[20] C. Seger, J. J. Joyce, *A Two-level Formal Verification Methodology using HOL and COSMOS*, Technical Report 91-10, Department of Computer Science, University of British Columbia, 1991.

[21] S. Wolfram, *Mathematica, A System for Doing Mathematics by Computer*, Addison-Wesley 1988.