VSDITLU: a verifiable symbolic definite integral table look-up

A. A. Adams, H. Gottliebsen, S. A. Linton, and U. Martin

Department of Computer Science, University of St Andrews, St Andrews KY16 9ES, Scotland
{aaa,hago,sal,um}@cs.st-and.ac.uk

Abstract. We present a verifiable symbolic definite integral table look-up: a system which matches a query, comprising a definite integral with parameters and side conditions, against an entry in a verifiable table and uses a call to a library of facts about the reals in the theorem prover PVS to aid in the transformation of the table entry into an answer. Our system is able to obtain correct answers in cases where standard techniques implemented in computer algebra systems fail. We present the full model of such a system as well as a description of our prototype implementation showing the efficacy of such a system: for example, the prototype is able to obtain correct answers in cases where computer algebra systems [CAS] do not. We extend upon Fateman's web-based table by including parametric limits of integration and queries with side conditions.

1 Introduction

In this paper we present a verifiable symbolic definite integral table look-up: a system which matches a query, comprising a definite integral with parameters and side conditions, against an entry in a verifiable table, and uses a call to a library of facts about the reals in the theorem prover PVS to aid in the transformation of the table entry into an answer. Our system is able to obtain correct answers in cases where standard techniques, such as those implemented in the computer algebra systems [CAS] Maple and Mathematica, do not. The importance of this work lies both in the novelty of verifiable table look up, and, more generally, as an indication of how theorem proving, particularly embedded verification with library support, can be a valuable tool to support users of mathematics, such as engineers, who want trusted results with minimal user interaction. NAG Ltd, the developers of the CAS **axiom**, brought this problem to our attention and are interested in including such a system in future projects.

Tables of mathematical formulae have been used by engineers and technicians for centuries. Inevitably such tables contained errors, sometimes slips of the pen, sometimes deliberate changes to foil copyists. In many cases accessible high-speed computation has allowed us to replace tables with on-the-fly calculation. For example a navigator's instruments and tables can now be replaced by an efficient GPS device, or an engineer's handbook with an "interactive book" based on a computer algebra system. However they are not entirely obsolete: the notorious Pentium bug was due to an error in a look-up table for SRT division, and this prompted the development by Owre and others of a general framework in the PVS prover for handling such tables [ORS97].

Definite integration, or "finding the area under a curve" is traditionally carried out using numerical techniques. However these cannot be used in the presence of parameters, and it is widely recognised in the computer algebra community [Sto91,Dav] that symbolic definite integration in the presence of parameters is a tricky problem where current algorithms are not adequate and computer algebra systems can get even very simple examples wrong. Thus table-look up is recognised as a useful solution, particularly as the answers often contain subtle side conditions. Machine look-up tables have obvious advantages over paper ones, offering automated pattern matching and simplification, ability to handle far more complex table entries and side conditions, and interoperability with other software. In particular web-based tables can be routinely updated with new results, and allow sharing and reuse of entries which may be complicated to obtain and are likely to be useful to other practitioners, as well as providing interesting opportunities for investigating user demand (which is often for nothing more difficult than homework problems). All the published paper look-up tables contain errors, so verifiable machine look up tables offer a greater possibility of freedom from error through the use of machine certification of the table entries, and their verifiable transformation to a correct answer.

In the next section we describe the problem of symbolic integration in more detail, and indicate why look-up tables are valuable. Section 3 describes the full concept for a VSDITLU, while Section 4 describes our prototype implementation, which is able to obtain correct answers in cases where standard techniques, such as those implemented in Maple and Mathematica, do not. Our prototype system extends the best available electronic table, Fateman's web-based table [EF95], by including parametric limits of integration and queries with side conditions. Section 5 addresses some of the wider issues and places our work in the context of other recent work on integrating theorem proving and computer algebra.

2 Symbolic definite integration

Thirty years ago an engineer wishing to compute a standard indefinite or definite integral, and preferring a trusted authority over uncertain high school math skills, would have turned to books of tables such as Gröbner and Hofreiter [GH61] or the CRC tables [ZKR96]. For example in the four hundred or so pages of [GH61] we find Entry 7 of table 1 Volume I, which expresses an indefinite integral, or more precisely an antiderivative[1], in high school math terms "a function whose derivative is $(ax^2 + 2bx + c)^{-1}$"

$$\int (ax^2 + 2bx + c)^{-1}\, dx = \tag{1}$$

[1] Note that throughout this paper, for a a positive real, $\sqrt{a}$ denotes the positive square root of a and $\mathrm{Log}(a)$ denotes the natural logarithm of a. Log denotes the principal value of the complex logarithm function.

$$\begin{array}{ll} \dfrac{\mathrm{Log}\,|(ax+b-p)/(ax+b+p)|}{2p}+D & \text{for } p=\sqrt{b^2-ac}, ac<b^2 \\ \dfrac{\tan^{-1}((ax+b)/p)}{p}+D & \text{for } p=\sqrt{ac-b^2}, ac>b^2 \\ -\dfrac{1}{(ax+b)}+D & \text{for } ac=b^2 \end{array}$$

and Entry 2 of Table 13 Volume II, which expresses a definite integral, in high school math terms "the area under the curve $(ax^2+2bx+c)^{-1}$ for x between 0 and 1":

$$\int_0^1 (ax^2+2bx+c)^{-1}\,dx = \tag{2}$$

$$\begin{array}{ll} \dfrac{\mathrm{Log}\,|(b+c+p)/(b+c-p)|}{2p} & \text{for } p=\sqrt{b^2-ac}, ac<b^2, c\neq 0 \\ \dfrac{\tan^{-1}((a+b)/p)-\tan^{-1}(b/p)}{p} & \text{for } p=\sqrt{ac-b^2}, ac>b^2 \\ a/b(a+b) & \text{for } ac=b^2>0, \text{and } b/a>0 \text{ or } b/a<-1 \end{array}$$

To evaluate, say

$$\int_0^1 (x^2+2\sin(d)x+1)^{-1}\,dx \text{ for } |d|<\pi/2 \tag{3}$$

the user matches $a=1, b=\sin d, c=1$, observes that as $1>\sin^2 d$ the second case alone applies, simplifies under the constraint $|d|<\pi/2$ the expression in $\sin, \tan^{-1}$ that results and obtains the result $(\pi-2d)/4\cos d$. No understanding of integration, limits, singularities of the integrand and so on is involved, just symbol manipulation to render the answer in an acceptable form.

In each case the table entry gives an indefinite or definite integral with respect to x of an integrand involving real parameters a, b and c, with limits of integration $0, 1$ in Entry 2, together with certain side conditions on each answer involving the parameters. Entry 1 is complete: the side conditions partition all possible values of a, b, c. Entry 2 is not complete: it does not cover, for example, the case $c=0$, where in fact the integral is undefined. The entries above are correct: in 1968 Klerer and Grossman [KG68] showed that all current tables contained a small number of errors, mostly typographical: for example [ZKR96] contains a sign error in a version of Entry 1. Notice that while the left hand sides of equations such as (1), (2) are well-defined functions they may have a wide variety of representations: there is no useful notion of canonical form here and so there are in general many possible ways of expressing the table entries.

Attempts have been made to produce electronic versions of such tables, for example the CRC Standard Math Interactive CD [Zwi98], but this still contains errors, and does not offer the facilities one might expect, such as automatic matching and simplification of integrals against user input, or exporting in a standard interchange format such as OpenMath [DGW97]: not surprising when several of the formulae seem to be stored only as images! There are at least two web based look-up tables: Fateman's [FE] handles

symbolic definite integrals with numeric limits of integration, includes all of the CRC entries without parametric limits of integration and is believed to be error free. The Mathematica table [Wol] is limited to indefinite integrals and calls Mathematica code which as we shall see often returns incorrect answers.

Symbolic integration algorithms inside computer algebra systems are very powerful, but as we shall indicate are currently not adequate for symbolic definite integration: hence the need for look-up tables. It is very easy for a naive user to get completely wrong answers, or get no answer at all, on input where high school techniques would find the answer fairly readily. For example even on indefinite integration Mathematica 3 returns

$$\int x^{-(a+1)/(a+1)}\, dx = \frac{x^{-(a+1)/(a+1)}}{-\frac{(a+1)}{(a+1)}+1} = x^{-1}/0$$

where the correct answer is $\mathrm{Log}(x)$. It returns

$$\int (x-b)^{-1}\, dx = \mathrm{Log}(x-b),$$

without adding the side condition $x > b$ or equivalently using the more familiar answer $\mathrm{Log}\,|(x-b)|$. It then evaluates the definite integral as

$$\int_0^c (x-1)^{-1}\, dx = \mathrm{Log}[c-1] - \mathrm{Log}[-1] = \mathrm{Log}[c-1] - i\pi \tag{4}$$

which gives the correct answer for $c < 1$ as the two imaginary numbers cancel out, but a complex number for $c > 1$, when the correct answer is the real number $\mathrm{Log}|c-1|$. For simplified versions of Entry 1 Mathematica 3 returns

$$\int \frac{1}{x^2-a}\, dx = -\frac{\tanh^{-1}(x/\sqrt{a})}{\sqrt{a}}. \tag{5}$$

without side conditions: in fact while $1/(x^2-a)$ is defined except where $x^2 = a \geq 0$, the right hand side of (5) is only defined over the reals for $0 \leq x^2 < a$, where it is equal to the expression involving $\mathrm{Log}(a)$ given in Entry 1. Called upon to evaluate

$$\int_{-1}^{1} \frac{1}{x^2-\cos(a)}\, dx$$

Mathematica 3 uses Equation (5) without taking account of the possible sign change of $\cos(a)$ to get completely wrong answers. Maple V [Hec96] performs similarly. Experts can set additional flags or write further code to avoid some of these problems, but this is not straightforward for the naive user.

A full explanation for these unexpected results and how to avoid them is outwith the scope of this paper: they are consequences of implementation compromises for what is,

despite the simple presentation given in high school, complex and subtle mathematics: see [Bro97,DST93]. Some would seem to be easily handled by greater care over calls to simplification routines, use of a type system such as in the computer algebra system **axiom** [JS92] or correct handling and propagation of pre- and side-conditions. Others are a consequence of problems in simplifying expressions in elementary functions[2], for which there is no canonical form or decision procedure, or in combining and simplifying parameterised expressions under constraints: for example the simplifications involved in Equation (3) defeated Maple and Mathematica.

A more fundamental problem involves the handling of functions over the reals and the blurring of computer algebra and computer analysis. CAS compute indefinite integrals by computing antiderivatives using the Risch algorithm, which involves decomposing the integrand as a sum of partial fractions. This can be expressed entirely algebraically through the theory of differential rings [Bro97]: rings with an operator satisfying $d(fg) = (df)g + f(dg)$. The Risch algorithm computes the antiderivative of a ring element f, that is an element g such that $dg = f$, generally over the complexes, where Log and $\tan^{-1}$ and so on are defined as the appropriate antiderivative: thus in this framework answers without side conditions may be correct.

There is no such framework for handling definite integrals, which are defined informally as "the area under a curve" and formally as a limit[3]. In high school we learn

$$\int_b^c f(x)\,dx = g(c) - g(b), \text{where } g \text{ is the antiderivative of } f \tag{6}$$

and this is the formula Mathematica 3 is using in the examples above. It returns incorrect results because (6) is false in general: by the Fundamental Theorem of Calculus it is true if f is defined and continuous in $[b, c]$ and there are straightforward modifications when f is piecewise continuous in $[b, c]$ or undefined at the endpoints of the interval: of course (6) may happen to give the right answer if none of these conditions is satisfied. Thus Mathematica gets (4) wrong because $1/(x - c)$ is discontinuous at $x = c$ where it has a pole (i.e. it "goes to infinity"), and (6) does not hold for $c > 1$. A correct symbolic definite integration procedure needs to work not only algebraically, as in the Risch algorithm, but analytically as well: treating poles, zeros and domains of definition of elementary functions such as Log and $\tan^{-1}$, and here computational techniques are far less well-developed. In particular since continuity is undecidable any algorithm must work with more tractable stronger pre-conditions: for example using a syntactic decomposition of the function to check for potential poles [Dup98].

This illustrates a more general design issue: there are many examples of processes, like definite symbolic integration via the Fundamental Theorem of Calculus, where a CAS may be able to compute an answer, sometimes correct, on a large class of inputs (any function where Risch returns an indefinite integral), be provably sound on only a subclass of those inputs (where the function is continuous) and be able to check soundness easily on a smaller subclass still (functions with no potential poles). Thus

[2] Elementary functions are those built up over the reals by combinations of log and exp, and include the usual trigonometric functions and their inverses

[3] For the purposes of this paper we assume the Lebesgue definition of integral.

the suppression of pre- and side-conditions is a design decision for ease of use. Some CAS such as **axiom** are cautious: only giving an answer when pre-conditions are satisfied. Others try and propagate the side conditions to inform the user, though this can rapidly lead to voluminous output. Mathematica and Maple generally attempt to return an answer whenever they can and leave to the user the burden of checking correctness.

The general problem of implementing symbolic definite integration is hard, calls upon many issues in the foundation of analysis and algebra, and involves both calculation (for example the algebraic factorisation and simplification required in the Risch algorithm) and proof (to verify the precondition to (6)). A fully verifiable implementation would involve a fully verified implementation of the major part of a computer algebra system, as well as several graduate level textbooks such as [Bro97].

Davenport [Dav] has described a general plan for symbolic definite computation: roughly speaking this analyses the function for possible poles, uses these to decompose the range of integration into components on which the integrand is continuous, applies the Risch algorithm and the Fundamental Theorem of Calculus on each component, then combines and simplifies the answers, all in the presence of parameters. We are currently implementing this using a combination of symbolic computation and theorem proving and expect better results than would be presently possible using computer algebra techniques alone. However this approach, while it is an exciting challenge for theorem proving research, seems hard to fully automate in general, and does not really solve the problem of our putative engineer who wishes to replace paper tables with a reliable automatic machine service, usually handling different instantiations of common integral schema. Thus we are led to consider the problem of a verifiable symbolic definite integral table look-up, which validates particular table entries and hence bypasses the difficulty of verifying the integration algorithm or calculating the correct definite integral.

Indefinite integration, the fundamental theorem of calculus and so forth have been developed in several theorem provers as part of a development of theories of real analysis, for example AUTOMATH [dB80], Mizar [Try78], HOL-light [Har98] and PVS [Dut96]. However such a development does not generally enable us to calculate anything but the simplest integrals. Harrison and Théry [HT94] experimented with combining such a development with the use of a computer algebra system as an oracle to compute indefinite integrals which were then verified by the prover. However this only helps if the computer algebra system gets the integral right!

3 The VSDITLU

We describe the principle of the VSDITLU, and our prototype implementation, and discuss the theorem proving tasks it generates. These can be characterised as

- validating the table entries, which is generally an expert interactive theorem proving task
- matching a query against the table
- verifying side conditions to return a result, generally an automated theorem proving task

We are considering expressions of the form

$$\int_b^c f(x, p_1, \ldots, p_n)dx$$

where x is a real variable, $b, c, p_1, \ldots, p_n$ are real [4] parameters, and f is a function over the reals.

The VSDITLU comprises a table of validated entries of the form

$$\int_b^c f(x, p_1, \ldots, p_n)dx : K, C \tag{7}$$

where $f(x, p_1, \ldots, p_n)$ is the integrand and K is a sequence of pairs of the form $\langle A, R\rangle$. Informally such a pair denotes that, under the constraints or side conditions R,

$$\int_b^c f(x, p_1, \ldots, p_n)dx = A,$$

while C records information to assist in verifying the table entry. More precisely A is a real expression, or "unknown" or "undefined", R is a boolean combination of equalities and pure inequalities over $\{b, c, p_1, \ldots, p_n\}$ and C is a certificate, a set of assertions for use in validating the entry. Formally the table entry asserts that for all values of the parameters b, c, p_i, and for each $\langle A, R\rangle$ in K we have

$$C \wedge (R \Longrightarrow \int_b^c f(x, p_1, \ldots, p_n)dx = A).$$

A table entry is said to be complete if it covers all possible values of the parameters, that is to say $\{\overline{R} \mid \langle A, R\rangle \in K\}$ partitions the parameter space, where $\overline{R}$ denotes the solutions of R. As we indicated above even complete table entries need not be unique. We discuss below the validation of table entries.

Figure 1 shows a typical table entry, omitting the certificates C.

To use the table the user submits a query

$$\left(\int_{b'}^{c'} g(x, p'_1, \ldots, p'_n)\ dx, Q\right), \tag{8}$$

where $b', c', p'_1, \ldots, p'_n$ are real valued parameters and Q is a boolean combination of equalities and pure inequalities over $\{b', c', p'_1, \ldots, p'_n\}$. The integral is matched automatically against the integrand of one or more table entries of the form (7) to obtain a

[4] In principle we could include additional type constraints such as integer or rational.

$$\int_b^c \frac{1}{p+qx}\ dx = \begin{cases} \underline{Answer} & \underline{Constraints} \\ \hline 0 & (b=c) \\ \hline \textit{undefined} & (q\neq 0)\wedge(b\neq c)\wedge((b=-\frac{p}{q})\vee(c=-\frac{p}{q})) \\ \hline \frac{\text{Log}\lvert qc+p\rvert-\text{Log}\lvert qb+p\rvert}{q} & (q\neq 0)\wedge(b\neq c)\wedge(b\neq -\frac{p}{q})\wedge(c\neq -\frac{p}{q}) \\ \hline \frac{c-b}{p} & (b\neq c)\wedge(p\neq 0)\wedge(q=0) \\ \hline \textit{undefined} & (b\neq c)\wedge(p=0)\wedge(q=0) \end{cases}$$

Fig. 1. A Typical VSDITLU Entry

match, or more generally a set of matches, Θ. We discuss the matching in more detail below.

So for example, a user might enter the query:

$$\left(\int_l^m \frac{1}{cos(d)+2x}\ dx, (m>3)\wedge(l>3)\right)$$

which the VSDITLU should match against the table entry in Figure 1 with $\phi = \{cos(d) \leftarrow p, 2 \leftarrow q, l \leftarrow b, m \leftarrow c\}$.

Having obtained the matchings, Θ and Q are used to return an answer

$$\int_{b'}^{c'} g(x, p'_1, \ldots, p'_n)\ dx : L$$

where L is a set of pairs of the form $\langle A', R'\rangle$, for A' a real expressions and R' a set of constraints. This denotes that for all values of the parameters b', c', p'_i, and for each $\langle A', R'\rangle$ in L we have

$$(R' \wedge Q) \Longrightarrow \int_{b'}^{c'} g(x, p'_1, \ldots, p'_n)\ dx = A'.$$

To solve for L we note first that for any $\langle A, R\rangle \in K$ and $\theta \in \Theta$ we have, for all values of the parameters b', c', p'_i, that

$$R\theta \Longrightarrow \int_{b'}^{c'} g(x, p'_1, \ldots, p'_n)\ dx = A\theta,$$

and hence

$$R\theta \wedge Q \Longrightarrow \int_{b'}^{c'} g(x, p'_1, \ldots, p'_n)\ dx = A\theta \tag{9}$$

and so if K is complete we may take

$$L = \{\langle A\theta, R\theta \wedge Q\rangle \mid \langle A, R\rangle \in K, \theta \in \Theta\}.$$

However if for some θ and R the set of constraints $R\theta \wedge Q$ has no solutions in b', c', p'_i, that is if

$$\neg\exists b', c', p'_i\ .\ R\theta \wedge Q, \tag{10}$$

then $\langle A\theta, R\theta \wedge Q\rangle$ contributes no extra solutions to L, and so we may assume

$$L = \{\langle A\theta, R\theta \wedge Q\rangle \mid \langle A, R\rangle \in K, \theta \in \Theta, (\exists b', c', p'_i\ .\ R\theta \wedge Q)\}.$$

Thus each $\langle A, R\rangle \in K$ gives rise to a possible side condition (10), and if this side condition can be proved $\langle A, R\rangle$ does not contribute to L. If the side condition cannot be proved even though the assertion is true, then $\langle A\theta, R\theta \wedge Q\rangle$ remains in L but is redundant.

In our example we have five such side conditions:

$$\begin{array}{l}
\neg\exists l, m, d.[b = c]\phi \wedge l > 3 \wedge m > 3,\\
\neg\exists l, m, d.[(q \neq 0) \wedge (b \neq c) \wedge ((b = -\frac{p}{q}) \vee (c = -\frac{p}{q}))]\phi \wedge l > 3 \wedge m > 3,\\
\neg\exists l, m, d.[(q \neq 0) \wedge (b \neq c) \wedge (b \neq -\frac{p}{q}) \wedge (c \neq -\frac{p}{q})]\phi \wedge l > 3 \wedge m > 3,\\
\neg\exists l, m, d.[(b \neq c) \wedge (p \neq 0) \wedge (q = 0)]\phi \wedge l > 3 \wedge m > 3,\\
\neg\exists l, m, d.[(b \neq c) \wedge (p = 0) \wedge (q = 0)]\phi \wedge l > 3 \wedge m > 3.
\end{array}$$

Of these the second, fourth and fifth are true, essentially as $-1 \leq \cos(d) \leq 1$, and so we obtain the answer

$$\int_l^m \frac{1}{cos(d) + 2x}\ dx = \begin{cases} \underline{Answer} & \underline{Constraints} \\ 0 & (l = m > 3) \\ \mathrm{Log}\dfrac{|2m + cos(d)|}{|2l + cos(d)|} & (l \neq m) \wedge (l > 3) \wedge (m > 3) \end{cases}$$

Notice that for concision we have simplified the constraints $R\phi \wedge Q$ remaining in the answer. As before there is no canonical way to do this, though there are sometimes obvious redundancies and subsumptions to be eliminated.

We now discuss the theorem proving tasks in more detail. Rather than work with a precise class of functions (for example linear or polynomial), when we could develop a clear theoretical analysis of the scope and limitations of our methods, we have deliberately put no restrictions on the real functions that can occur in the table entries. Matters are undecidable in general, and we necessarily can only give a rather vague account of the scope of our techniques, trusting rather on implementing a range of methods which can cover tractably the sort of input that is likely to occur in practice (not too deeply nested for example).

Validating the table entries

To add a new entry to the table it is necessary to provide K and to verify that the entry is correct. There are two parts to this verification:

- showing that the entry (7) is correct
- showing that the entry is complete

In the previous section we described what would be involved in computing and verifying an integral from scratch via the Risch algorithm. What we propose here is more straightforward: we assume that the computation has already been done, possibly by ad hoc means or by calling upon an existing table, and all that is required is to validate the result. The certificate C allows us to provide auxiliary lemmas to assist in this. Thus for example the part of the certificate covering the second line of (2) is just the assertion that $(ax^2 + 2bx + c)^{-1}$ is continuous in $[0, 1]$. To verify this part of the entry we need first to verify the indefinite integral of Entry (1) (which we can do by differentiating it), then to verify the certificate, and then we may invoke the Fundamental Theorem of Calculus (6) to verify the answer. Nonetheless it is still unlikely that except in the very simplest of cases this verification could be carried out automatically unless each certificate included a proof outline drawn up by a domain expert: the proof needs to call on a rich lemma database of facts about continuity, singularities, elementary functions and so forth.

We have not yet worked out a format for C suitable for a production version of the VSDITLU. It seems likely that proof planning [KKS96] will be useful here: the certificate might comprise a full proof plan or a standard template with information about poles and so forth for use with a pre-prepared proof plan.

The second part of this verification involves showing completeness by showing that $\{\overline{R} \mid \langle A, R\rangle \in K\}$ partitions the parameter space, where $\overline{R}$ denotes the set of solutions of R. If the number of cases is small this is a straightforward task for PVS, particularly if the constraints are linear and fall to its built in linear arithmetic package. We may reduce the theorem proving task by requiring only that $\{\overline{R} \mid \langle A, R\rangle \in K\}$ covers the parameter space, in which case we may have redundancy in our table entry, but it will still cover all cases. We have not yet addressed the problem of partial subsumption or overlap between different entries in the table: this comes back again to problems of representation.

Matching The most general form of matching here is undecidable: we are working over the reals and so for example $x + 2$ needs to match $b + x + 3$, $x - 1/b\, x + b^2$ and $x + 1/b^2$ but not $x - b^2$ or $x - 1/b^2$. The best we can hope for is a suite of methods sufficient to cover a wide range of cases: it is common also in computer algebra systems to make the problem more tractable by pre-processing functions to a standardised form, for example $x + a^2$ is represented as $x + c$ with the side condition $a^2 = c$. In addition certain forms, such as sums, tend not to occur in integral tables as it is assumed the user has pre-processed a query such as $\int (f(x) + g(x))dx$ into separate queries $\int f(x)dx$, $\int g(x)dx$. Note also that, while a query may match several entries, it is sufficient for our purposes to find a match against one complete table entry to get the required answer.

While there is a rich literature on matching and unification, as far as we know there is no existing implementation that is entirely adequate for our purposes. In his look-up table Fateman [EF95] uses pre-processing and stores the integrands in a particular kind of discrimination tree: matching is performed by a succession of approximate matches. Dalmas [DGH96], in his work on a deductive database of mathematical formulae uses a similar data-structure together with a conditional AC-unification algorithm implemented in ML (using logic programming techniques). Both systems appear to be correct but not complete, that is there are matches which they fail to find.

Proving the side conditions The side conditions have the form

$$\neg\exists b, c, p_i \ . \ H$$

where H is a boolean combination of equalities and strict inequalities involving real functions over $\{b, c, p_1, \ldots, p_n\}$.

For polynomial functions the problem may be addressed using quantifier elimination algorithms [Bro98] which solve the complementary problem

$$\exists b, c, p_i \ . \ H.$$

Extending these to other functions such as exp or log is currently an active research area, and in any case these methods are intractable for all but the smallest examples.

Thus we turn to theorem provers for reasoning about the reals. This can involve the development of substantial theories as found in textbooks, as was done in AUTOMATH [dB80] and Mizar [Try78]. However in practical applications such as this what is often needed is a library of more low level lemmas unlikely to be found explicitly in text books, such as $\forall x \ . \ 0 \leq cos^2(x) \leq 1$, together with a tactic mechanism which allows them to be applied automatically.

Harrison [Har98] developed a large portion of real analysis in HOL-light, both major theorems and also setting in place the mechanisms (power-series and so forth) to prove low-level lemmas about elementary functions. The reals are constructed by means of Dedekind cuts. By contrast Dutertre [Dut96] uses an axiomatic approach, extending the built in axiomatisation of the reals, to prove results about the reals in PVS, and again proves both major theorems and more low-level results. Fleuriot [FP98] has also implemented both classical and non-standard reals in Isabelle.

4 Our implementation

Our implementation consists of a front end comprising the table entries and a matching algorithm: around 2000 lines of Allegro Common Lisp. At present table entries and calls must be in a fairly strict standard form and we do not do any additional simplifications or redundancy checks: in principle our front end could be interfaced to a computer algebra system such as **axiom** for pre- and post-processing so as to handle a wider variety of inputs. The standard form aids the matching which is currently a basic form of AC pattern matching which makes no attempt to account for the units $(1, 0)$ of the AC operators $(+, *)$. For a full description see [AGLM99]. Some of the table entries

have been validated though we have not yet developed the notion of certificate very precisely.

PVS is called through emacs to prove the side conditions and return an answer. This is intended to be fully automatic: it uses a large lemma database of elementary facts about the reals, which we have built on top of Dutertre's implementation [Dut96], and uses the PVS "grind" command which applies a brute force search in an attempt to prove the required results. Initially we added properties of elementary functions as additional axioms on an ad hoc basis, but after experimenting with this we decided to re-implement Harrison's work in PVS to give a basis for proving whatever lemmas we need about elementary functions. Grind in turn is calling built-in PVS procedures to handle Boolean combinations and inequalities: if for example the inequalities happen to be linear a further efficient decision procedure can be called.

Development time was very short: 3-4 person months. Our choice of PVS was a fairly pragmatic one, based on what system had the best real library available in August 1998: more recent work in HOL makes it also a suitable candidate. Our table currently contains six entries and we have been able to evaluate correctly around 60 examples from a test suite of symbolic definite integrals that CAS running in a naive mode were unable to evaluate or got wrong.Our implementation got no answers wrong, but it did sometimes fail to return an answer because our matching algorithm was not powerful enough. On one occasion *grind* sent PVS into an apparently infinite loop and it was unable to identify an obvious counter-example to the non-existence of values for the parameters in a particular theorem.

Some of the integrals in the table are:

$$\int_b^c \frac{p}{x^2+a}\,dx\ (*) \qquad \int_b^c \frac{1}{x^2+a}\,dx\ (*) \qquad \int_a^b x\tan^{-1}(\frac{1}{x})\,dx$$

The integrals marked $(*)$ have a complete set of answers. No CAS that we tested (Maple V, Mathematica 3, **axiom** and Matlab) was able to consistently produce full correct answers to these integrals: in fact, all these CAS produce incorrect answers to some of them.

5 Discussion

The integration of computer algebra and theorem proving has attracted much research interest recently, in the form of verifying computer algebra algorithms in theorem provers [Thé98], adding inference mechanisms to CAS [CZ94], using proof planning techniques to aid in organising calculations [KKS96] or arranging for provers to make calls to CAS [HT94,HC94], either as oracles for results that are then verified or as trusted components for routine manipulations. We have argued elsewhere [Mar98] that while such endeavours are valuable as contributions to theorem proving research the resulting systems are not necessarily widely used by mathematicians as they fail to address issues of mathematical practice: how computational mathematics is actually done.

We have used automated reasoning tools to solve a problem identified by the computer algebra community, and which current computer algebra systems or look-up tables did not solve satisfactorily. Our system is designed for users of mathematics such as engineers, rather than expert mathematicians: such users want a black box system which solves the problem at hand rather than grappling with anything that requires user interaction with a theorem prover.

The success of this work leads us to suggest a way forward for the integration of computer algebra systems and theorem provers: theorem proving technology could be used to provide a variety of black box components for incorporation into applications like VSDITLU. Of particular interest would be components that enhanced, rather than duplicating, the capabilities of computer algebra systems. Theorem proving over the reals is a particularly important area as functions over floating point systems or the reals occur in a variety of engineering and safety critical applications: for example Dutertre's work was originally motivated by an avionics application. We have been using PVS as such a black box for determining if Boolean combinations of inequations and equations are solvable: other examples might include matching or unification engines for real functions, which would replace the somewhat ad-hoc matching algorithm that we implemented, engines for determining continuity of a function in a given region or engines for simplifying expressions in elementary functions.

Acknowledgements

We acknowledge support of the UK EPRSC under grant number GR/L48256 and of NAG Ltd, and we thank Mike Dewar from NAG for his interest and suggestions, James Davenport and Richard Fateman for advice on computer algebra, and Bruno Dutertre for allowing us to extend his code for the reals in PVS.

References

[AGLM99] A. A. Adams, H. Gottliebsen, S. A. Linton, and U. Martin. A Verifiable Symbolic Definite Integral Table Look-Up. Technical Report CS/99/3, University of St Andrews, 1999.

[Bri97] E. Brinksma, editor. *Tools and Algorithms for the Construction and Analysis of Systems (TACAS '97)*. Springer-Verlag LNCS 1217, 1997.

[Bro97] M. Bronstein. *Symbolic integration I*. Springer-Verlag, Berlin, 1997. Transcendental functions.

[Bro98] C. W. Brown. Simplification of truth-invariant cylindrical algebraic decompositions. In Gloor [Glo98], pages 295–301.

[Bun94] A. Bundy, editor. *CADE-12: 12th International Conference on Automated Deduction: Proceedings*. Springer-Verlag LNAI 814, 1994.

[CC94] J. Calmet and J. A. Campbell, editors. *Integrating Symbolic Mathematical Computation and Artificial Intelligence*. Springer-Verlag LNCS 958, 1994.

[CL96] J. Calmet and C. Limongelli, editors. *Design and Implementation of Symbolic Computation Systems, International Symposium, DISCO '96*. Springer-Verlag LNCS 1128, 1996.

[CZ94] E. Clarke and X. Zhao. Combining symbolic computation and theorem proving: Some problems of Ramanujan. In Bundy [Bun94], pages 758–763.

[Dav] J. H. Davenport. Really Strong Integration Algorithms. In preparation.

[dB80] N. G. de Bruijn. A Survey of the Project AUTOMATH. In Seldin and Hindley [SH80], pages 579–606.

[DGH96] S. Dalmas, M. Gaëtano, and C. Huchet. A Deductive Database for Mathematical Formulas. In Calmet and Limongelli [CL96].

[DGW97] S. Dalmas, M. Gaëtano, and S. Watt. An OpenMath 1.0 Implementation. In Küchlin [Küc97], pages 241–248.

[DST93] J. H. Davenport, Y. Siret, and E. Tournier. *Computer algebra*. Academic Press Ltd, London, second edition, 1993.

[Dup98] B. Dupée. Using Computer Algebra to Find Singularities of Elementary Real Functions. Available from the author, bjd@maths.bath.ac.uk, 1998.

[Dut96] B. Dutertre. Elements of Mathematical Analysis in PVS. In von Wright et al. [vWGH96].

[EF95] T. Einwohner and R. J. Fateman. Searching techniques for Integral Tables. In Levelt [Lev95], pages 133–139.

[FE] R. J. Fateman and T. Einwohner. TILU Table of Integrals Look Up. Web Service. http://http.cs.berkeley.edu/∼fateman/htest.html.

[FP98] J. Fleuriot and L. Paulson. A combination of nonstandard analysis and geometry theorem proving with application to Newton's Principia. In Kirchner and Kirchner [KK98], pages 3–16.

[GH61] W. Groëbner and N. Hofreiter. *Integraltafel*. Springer-Verlag, Vienna, 1961.

[Glo98] O. Gloor, editor. *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation*. ACM Press, 1998.

[Har98] J. Harrison. *Theorem Proving with the Real Numbers*. Springer-Verlag, 1998.

[HC94] K. Homann and J. Calmet. Combining theorem proving and symbolic mathematical computing. In Calmet and Campbell [CC94], pages 18–29.

[Hec96] A. Heck. *Introduction to Maple*. Springer-Verlag, New York, second edition, 1996.

[HT94] J. Harrison and L. Théry. Extending the HOL theorem prover with a computer algebra system to reason about the reals. In Joyce and Seger [JS94], pages 174–185.

[JS92] R. D. Jenks and R. S. Sutor. *AXIOM*. Springer-Verlag, 1992.

[JS94] J. J. Joyce and C-J. H. Seger, editors. *Higher order logic theorem proving and its applications*, Berlin, 1994. Springer-Verlag LNCS 780.

[KG68] M. Klerer and F. Grossman. Error Rates in Tables of Indefinite Integrals. *Journal of the Industrial Mathematics Society*, 18:31–62, 1968.

[KK98] C. Kirchner and H. Kirchner, editors. *CADE-15: 15th International Conference on Automated Deduction: Proceedings*. Springer-Verlag LNAI 1421, 1998.

[KKS96] M. Kerber, M. Kohlhase, and V. Sorge. Integrating Computer Algebra with Proof Planning. In Calmet and Limongelli [CL96].

[Küc97] W. W. Küchlin, editor. *Proceedings of the 1997 International Symposium on Symbolic and Algebraic Computation*. ACM Press, 1997.

[Lev95] A. H. M. Levelt, editor. *Proceedings of the 6th International Symposium on Symbolic and Algebraic Computation, ISSAC '95*. Springer-Verlag LNCS 1004, 1995.

[Mar98] U. Martin. Computers, Reasoning and Mathematical Practice. In Computational Logic, NATO Adv. Sci. Inst. Ser. F Comput. Systems Sci., 1998.

[ORS97] S. Owre, J. Rushby, and N. Shankar. Integration in PVS: Tables, Types, and Model Checking. In Brinksma [Bri97], pages 366–383.

[SH80] J. P. Seldin and J. R. Hindley, editors. *To H.B. Curry: essays on combinatory logic, lambda calculus and formalism*. Academic Press, 1980.

[Sto91] D. Stoutemyer. Crimes and misdemeanours in the computer algebra trade. *Notices of the AMS*, 38:779–785, 1991.

[Thé98] L. Théry. A Certified Version of Buchberger's Algorithm. In Kirchner and Kirchner [KK98], pages 349–364.

[Try78] A. Trybulec. The Mizar-QC 6000 logic information language. *ALCC Bulletin*, 6:136–140, 1978.

[vWGH96] J. von Wright, J. Grundy, and J. Harrison, editors. *Theorem Proving in Higher Order Logics: 9th International Conference*. Springer-Verlag LNCS 1125, 1996.

[Wol] Wolfram Research Inc. The integrator: the power to do integrals as the world has never seen before, http://www.integrals.com.

[ZKR96] D. Zwillinger, S. G. Krantz, and K. H. Rosen, editors. *CRC standard mathematical tables and formulae*. CRC Press, Boca Raton, FL, 30th edition, 1996.

[Zwi98] D. Zwillinger. Standard Math Interactive. CD-ROM, 1998.