A Proposal for Automated Integral Tables (Work in Progress)

Richard J. Fateman
Theodore H. Einwohner

March, 1992

Abstract

One of the long-term general goals of algebraic manipulation systems has been the automation of difficult or tedious, yet common, symbolic mathematical operations. Prominent among these has been symbolic integration. Although some effective algorithms have been devised for integration especially for those problems solvable in terms of elementary functions and a few additional special functions, the vast majority of entries in large tables of indefinite or definite integrals remain out of reach of current machine algorithms. We propose techniques for introducing the information in such tables to computers, extending such tables, and measuring the success of such automation.

Similar tabular data concerning simplifications, summation identities, and similar formulas could also be treated by some of the same techniques.

1 Introduction

One of the long-term goals of algebraic manipulation systems has been the automation of symbolic mathematical computations which are widely useful but are difficult or error-prone for humans. Among the earliest domains considered by computer algebra system builders were differential and integral calculus. If we ignore the issue of presenting the answer in simplified form, explicit differentiation turns out to be simple to program[1]. Integration is more difficult.

1.1 Indefinite Integration

In the 1960's early workers on symbolic integration [29], [22] limited their problem domain to *indefinite integration problems* as presented by problems in elementary calculus texts. This domain restriction (which was probably not noticed by Freshman and their instructors) made it appear that the integration problem had been either solved, or nearly solved.

But not all problems of interest are solvable by such naive methods, and the emphasis on symbolic *indefinite* integration research in subsequent years was to clearly define and extend, step-by-step, the class of "elementary" expressions which could be integrated "in closed form in terms of elementary functions" within the context of the Liouvillian "problem of integration" [27] [6] [20].

These techniques are a far cry from the very simple heuristic called "derivative divides"[22] which can solve some 80% of standard homework problems.

Various other methods have developed for the other 20% of homework problems: These include techniques for rational function integration (via partial fractions), heuristic transformations [29], specialized pattern-matching programs [22], and (eventually) Risch's solution to Liouville's conjecture via differential algebra [27]. Combined methods provide impressive demonstrations of success in this domain. (See Norman [24] for a discussion.)

[1]So simple that in appropriate languages such as Lisp, SNOBOL, or Prolog, differentiation can be used as an elementary programming exercise.

Combinations of the techniques for elementary indefinite integration are included in computer systems such as Axiom, Derive, Macsyma, Maple, Mathematica. These programs are generally considered useful even if the set of solvable problems falls very far short of "all integral calculus".

Can the same programs in these systems solve the occasional *challenging* problems that, for example, are presented in problem sections of the *SIAM Review*? Davenport [8] claimed that at some intermediate level of difficulty, the Axiom system[2] with probably the most ambitious of the integration sub-systems, was of some use, but was still not a panacea.

Some of the failures one can cite are related to the lack of implementation of known decision procedures which continue to require subtle techniques (for example, Bronstein's work [4], implemented in Axiom), and some require mathematical extensions of the Liouville-Risch style decision techniques [6], [7], [3], etc. Often failures are produced because a system does not support a sufficiently thorough analysis of (say) zeroes of expressions, misinterpretes multi-value functions, or lacks adequate algorithms for factorization over algebraic extensions. Often simplification routines are not powerful enough. Yet even if we look in this direction for solution to a wide range of problems outside the realm of calculus texts, the algorithmic approach seems to falter. Especially if we include definite integration, there is enormous variety and not enough well-understood methodology to provide a simple and uniform algorithmic approach.

Furthermore, even if we were to extend the algorithms by better implementation, most users of systems cannot make use of an answer that says "the integral does not exist in closed form in terms of 'elementary' functions" (e.g. exp, log, trig, arctrig, rational roots). Most users would prefer an answer, expressed if necessary in terms of "special functions" or perhaps as a table of approximations which might be symbolic or numeric. A response such as "this system is unable to evaluate the definite integral $F(a, b)$ symbolically by algorithmic or heuristic methods, but here's a numerical procedure which will provide an answer to any desired error...." may be appropriate. For mathematicians searching for a symbolic answer, additional appropriate answers might be, "I have some expensive transformations I could try, and then attempt the integration again. Shall I?" or the program might respond with the presentation of "nearby" expressions which it can integrate, suggesting that by some human cleverness or physical approximation, the integrand could be transformed to the nearby one.

1.2 Definite Integration

There is a large class of definite integrals that can be done by simply computing the indefinite integral and substituting upper and lower bounds. Freshman calculus students may excellent use of this technique.

Unfortunately, much more subtle techniques are needed for the class of integrands that cannot be handled this way. Although the history of attacks on this problem goes back to Wang [32] on automatic contour integration and Avgoustis [2] on use of special functions, the overall success rate is considerably lower on this challenge.

Relevant to the definite integration problem, we must point out that the indefinite integration programs are, for the most part, actually algebraic anti-differentiation (including all those based on the Risch [27] decision procedure) and they do not guarantee the analytic validity of their answers anywhere. In fact, such programs routinely produce numerically unstable formulas [12], as well as outright incorrect answers caused by incorrect interpretation of branch cuts.

Consider the following example [21]. If we differentiate the expression

$$f = \arctan(\frac{x^3 - 4x}{x^2 - 1})$$

we find the answer is

$$d = \frac{df}{dx} = \frac{x^4 + x^2 + 4}{x^6 - 7x^4 + 14x^2 + 1}.$$

A sufficiently powerful algorithmic attack on algebraically integrating the rational function d could hardly do better than f. (Some systems require some additional simplification step to get this form).

Therefore, one might falsely conclude that we could compute the integral of d, with respect to x (note that d is an expression entirely free of zeros or poles) from -2 to 2 by substituting the values -2 and 2 into f, and

[2] then called Scratchpad-II

computing the difference. Those values are each 0, so the integral computed this way would be 0, even though d is strictly positive. (Proof that d is always positive and finite for real finite x: Algebraically, the numerator of d is clearly positive, and the denominator is expressible as a sum of squares: $(x^2-1)^2+(x^3-4x)^2$.) A numerical quadrature program tells us the area under the curve is remarkably close to 2π. To understand what when wrong, you must notice that the arctan function is discontinuous at -1 and 1, although bounded in the range of integration. In fact, a correct choice of branches for the arctan function provides the exact answer 2π.

2 What are we trying to compute?

Let us assume now that our domain is no longer limited to Freshman calculus problems. Our intent is to provide an automatic integration program suitable for use by applied mathematicians. It should make hard-copy tables of integrals as obsolete as 10-place logarithm tables are today. Furthermore, entries which are not directly in any table, but yield to known algorithms, should be available. When appropriate, bounds on regions of validity should be announced, and applied when possible, etc. The algorithms should be available to illustrate the methods used and allow for improvement. This is a formidable challenge.

2.1 Indefinite integrals

One large widely-distributed table [16] has 1160 pages of formulas, of which about 150 pages are indefinite integrals of elementary functions (generally expressed as elementary functions). Some, but perhaps not even the majority of these elementary integrals can be done algorithmically by methods incorporated in most computer algebra systems. Of course many untabulated integrals, especially ones that are messy (but routine) can be done by computer systems, but are difficult to compute by hand[3]. The overlap between the algorithms and the tables is somewhat overestimated by the 150 pages because some sets of entries can be derived from a single "general" entry by substitution. For example, formulas 2.644.1 through 4 cover the next 9: 2.644.5 through 14 in [16]).

Interestingly, Gradshteyn's table of *indefinite* integrals of special functions, is only 7 pages long. This suggests that the effort to finding further extensions of the Risch algorithm for additional indefinite integrals involving special functions may not be highly fruitful, (or perhaps that the explorations of the past have not found such information useful).

2.2 Definite integrals

Gradshteyn's table includes some 400 pages of *definite* integrals of elementary functions (with parameters) many of which use special functions in their answers, and some 280 pages of definite integrals of special functions. Prudnikov *et. al.* [26]have even more definite integrals. Few of these integrals can be done by any of the algebra systems, although a few classes of them can be attacked by specific methods [25], [15], [32]. A recent paper by Adamchik and Marichev [1] suggests that many (perhaps 80%) of the integrals in what is probably the largest published table of integrals [26] can be mechanically produced by using tranformations into hypergeometric-type functions.

Even if this method is so successful, there are still substantial additional domains that still do not yield to uniform algorithmic methods, and so we continue on our quest to figure out how to access patterns.

2.3 An aside on expressions vs. programs

As a side comment, we have argued [12] that even the best of programs may produce answers which are not quite right. In fact, we believe that the emphasis on expressions, rather than computational schemes, is inherently dangerous. A typical dangerous formula is

$$\int_a^b \frac{1}{x}dx = \ln b - \ln a.$$

[3]for example, differentiating $\exp(\exp(x))\log(x)$ a few times is messy but algorithmically routine to integrate.

Better numerical formulas include $\ln(b/a)$, or less obviously, $2\,\mathrm{arctanh}\,((b-a)/(b+a))$. The latter formula is superior if b/a is outside the interval [1/2, 2]. The most useful answer to this integral may be a program something like this (this is in the syntax of the Macsyma system):

```
dlog(x,y):= /* compute log(x) - log(y) fast and accurately */
  if (x/y < 0.5) or (2 < x/y) then log(x/y)
        else 2*atanh((x-y)/(x+y)).
```

192z The arctanh function expression can be derived through computing a divided difference of the logarithm. This re-phrasing, and similar forms for other commonly encountered results of integration programs, are worth pursuing [12] as alternatives to the kinds of simplified representations used now for the symbolic program output.

Even given our bias for programs rather than expressions, for the rest of this paper let us return to the domain of expressions, assuming that, if necessary, we will be able to convert expressions into stable, efficient programs.

3 How do we compute integrals?

Although it is not entirely fair to do so, we will assume that the technology of the past 25 years in "antidifferentiation" is freely available. There are several generalizations and tricks that should be used, even in existing programs, but probably are not.

3.1 Heuristics and Tricks

1. Some integration problems are easy to solve if you merely remember, when you compute a derivative, how it came about. If the integration is merely one step in a longer calculation done by an algebraic manipulation program, this is sometimes useful.

2. Simple tricks like the derivative-divides technique, work much better when division is actually available, or perhaps factorization is used to deduce factors. Moses' SIN program [22] as implemented in Macsyma, uses only heuristic techniques based on pattern matching to determine exact division. A two-page Lisp program [13] we've written using canonical rational forms and actual division (rather than pattern matching) is substantially shorter and more effective than the heuristic version in Macsyma.

3. Knowledge about integration of functions should be distributed with the functions, and to the extent possible, not embedded in the integration program. Simple anti-derivatives can be stored with the functions; integrands involving combinations of functions should be stored in an indexed table (discussed further below). Certainly the integration program should be able to refer to derivatives supplied by the user, as necessary.

4. A simple interface to user-anticipated integrals should be provided. That is, there should be a mechanism to insert integration patterns that are likely to occur in a computation in the integration facility.

5. Known algorithms should be implemented and available. This is obviously true, but advice not widely followed simply because the Risch algorithm and its elaborations are rather subtle. It is also the that the Risch algorithm's version of an anti-derivative is sometimes quite far from the most useful result.

6. A table lookup system should be provided. Two questions immediately come to mind: (a) what should be stored? (b) how should information in the table be retrieved?

   If we consider table-lookup seriously, we can consider a program which, running decoupled from any specific problem, simply generates classes of solved integrals and stores the answers in a table. Especially as storage costs have been dropping rapidly for years, the question arises: Why compute a "custom integral" when one can be obtained "off the shelf"?

We have three objectives in the remaining parts of this paper. The first is to propose a number of techniques drawn from hash-coding, pattern matching, and database systems, for introducing large tables into a computing system. The second objective is to lay out a program by which the entries in at least the indefinite integration portion of a table of not-necessarily elementary functions can be provided, correctly, at relatively high speed. Our third objective is to propose a number of measures which can be used to judge the success of such automation.

Although we will not address this issue further here, similar considerations play a role in other databases of concerning simplifications or identities.

3.2 Tables of integrals

The vast majority of entries in large tables of indefinite or definite integrals (as well as indefinite and indefinite summations), remain out of algorithmic reach. This is not to say that algorithms cannot be produced for each entry in a table, but merely that, once we have used the usual "front-end" routines, the remaining integrable forms constitute a very small subclass of possible expressions. If those forms can be described economically by one or two patterns, then an elaborate algorithm is difficult to justify from practical grounds.

There are special considerations in looking up integrals. Consider the nature of patterns: The variable of integration (typically x) along with operators such as sin are "constants" and all other names (typically a, b, $\cdots$) are "variables" to be bound in the course of the match. Virtually no patterns are sums of expressions, simply because the integral of a sum can generally be computed as the sum of the integrals of each term. The patterns are generally products, quotients, or special functions.

3.2.1 Preprocessing of expressions

Most tables have patterns that are as general as possible (with many parameters). Many parameters will have to be matched to identities, and some parameters can be matched in a variety of ways corresponding to (for example) 1^1 or a^0 matching 1 (for suitable a). Some patterns will be evident only after performing certain transformations such as completing squares, rationalizing denominators, solving linear systems of equations, etc. Any conscientious table-builder should make some effort to remove what Gradshteyn [16] refers to as synonyms. For example, [16] page xxiii, considers the expression

$$\int_0^{\pi/4} \frac{(\cot x - 1)^{p-1}}{\sin^2 x} \ln(\tan x) dx = -\frac{\pi}{p} \csc(p\pi). \tag{1}$$

"By making the natural substitution $\cot x - 1 = u$ we obtain

$$\int_0^{\infty} u^{p-1} \ln(1+u) du = \frac{\pi}{p} \csc(p\pi). \tag{2}$$

Integrals similar to formula (1) are omitted in this new edition."

Conversion from form (1) to (2) would seem to be computationally straightforward given such a relatively simple substitution equation, and at least two computer systems (Macsyma and Maple) have an explicit `changevar` command to indicate this transformation[4]. In general there is an opportunity for complications in the process of change of variables in the case that inverting the transformation requires a choice from a set of multiple-valued inverses; and in the subsequent simplifications that are needed to show that the change of variables actually produced a simpler expression.

If this substitution is, as claimed, "natural" it should be possible to program a computer to try this choice, perhaps among others, as a potential transformation to reduce the depth of nesting of operators involving x in such an integral. The solution and substitution may require substantial computation. Thus it is not obvious if this is a pre-processing step to be tried early in a search for a solution, or if it should only be tried after

[4]Not that they necessarily get the same result. Version 2.0 of Mathematica seems unequipped to handle this task automatically, and is unequipped to simplify the expressions generated. Maple V also did not automate needed simplifications. Only Macsyma produced the full transformation becnause it used the identities (modulo multiple-valued inverses), $\tan(\operatorname{arccot}(1+u)) = (u+1)^{-1}$ and $\csc(\operatorname{arccot}(1+u)) = \sqrt{1+(1+u)^2}$. Macsyma produced warning message that multiple-valued inverse functions were being used: in fact, given the range of integration, one can show the message is superfluous.

failure of other methods. Usually a Slage [29] used a hill-climbing strategy and Moses[22] used a substitution approach more centrally in their integration programs. In a highly parallel computing environment more variations, including trying several simultaneous variable changes or other search strategies and heuristics could be tried.

3.2.2 General approaches

Strangely enough it is easy to forget very general strategies: The integral of the derivative of an (otherwise undefined) function $f'(x)$ is $f(x)$, for example.

Another (apparently neglected) heuristic is this quite simple one: $\int_{-a}^{a} f(x)dx = 0$ where f is an odd function. While Macsyma has applied this heuristic (probably since 1972), neither of the more recent systems Maple nor Mathematica seems to know it. The obvious test for a function being odd is that $f(x) + f(-x)$ must be zero for all $0 \leq x \leq a$. More generally, we could talk about f being "odd on an interval $[a, b]$" so that $\int_a^b f(x)dx = 0$ where $f(a + r) = -f(b - r)$ for all r such that $0 \leq r \leq b - a$.

This test for "oddity" can be tricky. Consider $\int_{-1}^{1} x^{-1}dx$. A system could easily conclude that $x^{-1} + (-x)^{-1}$ is identically zero and that the integral is 0. Unfortunately, the finiteness of $f(x)$ at $x = 0$ is problematical, and a more careful system would answer that the integral has *Principal Value* 0.

Additional general heuristics – that is, those that can be applied on integrals with nearly arbitrary functions f in them can be provided for various other symmetries, if they can be noted.

Corresponding to an observation of "evenness" on an interval, we can double the integrand and halve the interval.

For periodic functions, consider

$$\int_0^{2\pi} f(p\cos x + q\sin x)dx \;=\; 2\int_0^{\pi} f\left(\sqrt{p^2+q^2}\cos x\right)\,dx,$$

where f(x) is integrable on the interval $\left(-\sqrt{p^2+q^2}, \sqrt{p^2+q^2}\right)$.

And if we are going to match definite integrals, standardizing the limits of integration to just a few typical intervals may be useful. The most useful of the transformations may be

$$\int_a^b f(x)\,dx \;=\; (b-a)\int_0^1 f((b-a)t+a)\,dt$$

and

$$\int_a^b f(x)\,dx \;=\; (b-a)\int_0^{\infty} f\left(\frac{bt+a}{t+1}\right)(t+1)^{-2}\,dt$$

Additional "general" formulas involving arbitrary functions f in reductions – are cited by Gradshteyn [16] section 3.0.

3.3 Where do tables fit in with the algorithms?

Another question which arises is how a table-lookup approach should fit in with existing programs. The partial algorithmic methods programmed in most systems can handle many of the easy subcases of integration algorithmically, and this eliminates the need for certain classes of table entries, and can in principle surpass tables for specific algebraic, exponential and logarithmic cases, (At least those not involving reduction formulas).

In some cases, the "tables"actually specify algorithms or substitutions that may or may not be programmed already (e.g. an algorithm to transform rational functions of $\sin x$ and $\cos x$ to rationals functions of $\tan(x/2)$.) or may be covered by more general methods. (A more general method for this class of problems is to convert to complex exponentials and use the Risch algorithm. This happens to be a bad idea because the transformation is too "radical" and the answer may therefore seem unnecessarily complicated.)

Should we eliminate those the patterns that describe expressions we can integrate by algorithms? There is a good argument to leave at least some of them in. The tabulation of such "solved" problems can sometimes

provide "better" answers than the algorithms. In particular, some reduction formulas in a table may provide more insight that explicit forms for given expressions (e.g. $x^n e^x$).

This strategy assumes then that the tables are *not* being used as a last resort – only after the algorithms fail! The patterns could be used first, or in parallel. But then what if more than one answer is obtained? (Indeed one can sometimes get several correct answers by table lookup!) Should one return alternative answers to a single problem? Since it can be perplexing to students to compute the answer to $\int \sin x \cos x \, dx$ as $-(1/2)\cos^2 x$ while the answer in the back of the book is $(1/2)\sin^2 x$, this is plausible. In this particular case it would be hard to assert that one or the other of these two answers is simpler than the other. Observe, however that they are *not* identical, but differ by a constant.

And finally, what approach is likely to get the answer faster, more often?

We cannot predict this without a characterization of the workload, and an implementation of the algorithms, but we can make some observations. While the algorithmically solvable cases are perhaps the most "commonly occurring" integrals, including virtually every indefinite integral encountered in Freshman calculus, they represent a rather small percentage of the entries in the tables. Being most generous to the indefinite integration algorithms, (assuming, for example that they can produce answers in terms of elliptic functions) it seems they are likely to cover at most 150 pages in Gradshteyn and Ryzhik [16] out of over 1000. And as can be easily demonstrated for some of the programs, for some problems the "algorithms" produce wrong answers because they stumble across branch cuts.

3.4 Searching in integral tables

How should one search in the published tables? The scheme suggested by the organization of the major published tables (e.g. [16]) is to search for the occurrence of some prominent "outermost" most complicated function or subexpression, and then within some section, look for the presence of particular special inner forms [16]. One advantage to this approach is that –failing to find a successful match– one might be alerted to "nearby" integrable forms. This latter point may lead to a subtle approach likely to be difficult for computer programming.

We have been experimenting with a hash-coding that we hope could be applied to large integral tables (several hundred pages) This project has been simmering on a back burner since 1988, when a Berkeley graduate student (Michael Braverman) produced a term paper which demonstrated much faster time and a better "form" of the answer for an integration problem as compared to the Macsyma / Risch / definite integration system. Braverman's program (named Atheiʃt) was based on the use of a (potentially) very extensive table of integrals and a hash-code-based "characteristic vector" method of finding close matches between patterns and expressions. This tentative selection must be followed by a rather more elaborate unification of patterns and expressions to deduce the values of parameters in the pattern. As has generally been the case when attention is turned to a specific area of interest, this program has demonstrated substantial improvements in the "quality" of the answer, as well as a significant speed improvement when used on a subset of the domain it shares with Macsyma.

In continuing this work, the important issues to be addressed include

- the nature of indexing to be used to find probable matches,
- the nature of pre-conditioning expressions that might be needed for measuring closeness of match,
- the strenuousness of the matching process once some likely pattern is identified,
- the encoding of patterns as well as side-condition requirements so they can be applied constructively in the matching process,
- the design of the database so that it is inexpensive to use, and can be augmented without increasing the cost of matches. (See [18], [19] for some hints.)
- confirming the correctness of entries and/or generating new correct entries *ab initio*.

At first blush, it seems that tables of integrals are large data objects, and that storing patterns solely in main memory as Lisp symbolic expressions would be extravagant: large tables could affect the startup time for a system, or be a hazard in a paged time-shared environment when competing for resources. Furthermore, if we are going to generate new formulas, it seemed to us that to allow the table to grow to be encyclopedic, the data could not be just piles of patterns. Yet some back-of-the-envelope calculations suggest that even large printed tables (1000 text pages) could be encoded as uncompressed ASCII text in 10 megabytes, and perhaps much less. (1 megabyte!) Thus this particular problem domain does not seem to require some elaborate optical disk storage, or even much of a modern magnetic disk system. A (usually) read-only indexed file seems to fit the bill.

3.5 Nonsyntactic matching

Klier [18] has suggested the following technique. If you have reason to believe that a pattern and expression match, but need unify the two by finding the values parameters, a series expansion may sometimes help.

Consider a, b, and c in the pattern $P = a\sin^2 x + b\sin x\cos x + c\cos x$ which is alleged to match the expression $E = \sin(2x)$. Compute the Taylor series for $P - E$ about a convenient point (say 0), and solve the set of equations resulting from setting each of the coefficients to zero. In particular, the Taylor series begins:

$$-c + (-b+2)\ x + (c-a)\ x^2 + \frac{(2\,b-4)\ x^3}{3} + \cdots$$

from which we deduce that $a = 0$, $c = 0$, and $b = 2$. Hence (approximately, and in the case of a perfect match, exactly),

$$\sin(2x) = 2\sin x\cos x.$$

There are, of course, limitations to this approach, but it illustrates one alternative to syntactic pattern matching.

Milind Joshi, in his MS project at Berkeley [17] has written a pattern matcher that is externally similar to the Mathematica matcher [33], but is based on the premise that substantially more effort should be expended to try to match a pattern to an expression. For example, Mathematica declines to match the pattern x^{v-1} to x^5 (with $v = 6$). Joshi's system will, if necessary, attempt to simultaneously solve systems of linear or even non-linear algebraic equations to extract a match. Joshi's system furthermore has a pattern matching "critic" feature. A user who proposes a pattern may in fact be unaware of the boundaries of the match. While the pattern $ax + b$ would seem to describe the notion "linear in x" in fact, if b is allowed to match x^2, there may be some misunderstandings. The critic looks at plausible errors, proposes "boundary" conditions and suggests strengthening predicates.

Of course other systems each have their pattern matching approaches; Macsyma has no fewer than three: Moses' SCHATCHEN matcher from SIN [22] used for integration and summation, the user-visible matcher described in the manual through MATCH and TELLSIMP, and a third system (LETRULES) modeled closely on the successful pattern matcher in REDUCE.

The last word has not yet been said on ways to improve the incorporation of knowledge through patterns into symbolic mathematics systems. Pre-processing patterns to make them easier to match, compiling them in rule-sets, or otherwise attempting to avoid the combinatorial-explosion costs in searching, are useful directions to continue.

4 Generation of formulas

Verifying the correctness of the formulas in a large table is a non-trivial task. Even The data entry task of such a large mass of formulas, assuming they are correct, is also daunting. The restrictions on the domains of validity are also sometimes encoded in a non-effective way. Gradshteyn's view of domains is typical in this regard:

> If a particular formula is valid only for certain values of the constants (for example, only for positive numbers or only for integers), an appropriate remark is made *provided the restriction that we make does not follow from the form of the formula itself.* (section 2.0, emphasis added).

Deriving the restrictions in an appropriate and general form for such formulas from their presentation may not be trivial. Furthermore, computer algebra systems generally are not set up to manipulate such restrictions even if they were able to compute them (Some work at Berkeley by Adam Dingle [10] addresses some of the problems). Most systems are able to apply or simplify some restrictions when all the constants are explicit numerical quantities, but rarely can be relied upon to handle more complicated situations.

Following from the difficulty of verification is the hope that we can finesse the problem by mechanically deriving formulas (along with appropriate side conditions). Considering that a table of (at least indefinite) integrals could in principle be generated by generating and storing differentiations, this seems highly plausible at first glance.

4.1 Generating functions and *new* integrals.

If you consider the derivative of some class of random expressions, most will probably look like sums. (That is, their lead operator is '+'.) This is a simple consequence of the chain rule, the product rule and the sum r example with ule for differentiating.

On the other hand, integrals in a table will almost never be sums. The (usually) excellent reason for this is that integration is a linear operator: the integral of a sum is the sum of the integrals of the summands. Therefore the individual terms should be looked up separately.

As a consequence, differentiating randomly-generated expressions is not very attractive. We are working on techniques to make this more successful, but this is somewhat reminiscent, and perhaps less powerful than Piquette's work [25], also mentioned below.

4.1.1 Differentiating with respect to a parameter

Given an expression we can integrate in closed form $\int_a^b f(s,x)\ dx = F(a,b,s)$, but with parameters, we can generate additional integration equations by considering (under suitable conditions) the application of d/ds to both sides. As a simple example, consider the following integral of a Bessel function times an exponential ($s > 0$):

$$\int_0^\infty e^{-sx} J_0(x)\ dx = \frac{1}{\sqrt{1+s^2}}.$$

If we differentiate both sides with respect to s, we are presented with the (also valid formula):

$$\int_0^\infty -xe^{-sx} J_0(x)\ dx = -\frac{a}{(1+s^2)^{3/2}}.$$

Note that there is no difficulty evaluating the right-hand side for (say) negative values of s. The formula for the integral would be incorrect, however. The restrictions on s must be considered at every transformation.

This particular technique is especially useful for (Laplace, Fourier) transform style integrals where the formula for $e^{-sx}f(x)$ can be transformed to $x^n e^{-sn} f(x)$.

This process can be repeated indefinitely and its application (with appropriate differentiability and continuity requirements) is an easy (and apparently widely used) technique for generating table entries. (see also, for example Geddes [15])

4.1.2 Integrating with respect to a parameter

Consider a similar example – subject to well-behaved integrals on both sides, we can start with

$$\int_0^\infty e^{-sx} J_1(x)\ dx \ = \ 1 - \frac{s}{\sqrt{1+s^2}.}$$

Then by integrating both sides with respect to s we can generate a formula for

$$\int_0^\infty \frac{e^{-sx}}{x} J_1(x)\ dx \ = \ s - 1/2 \log\left(s^2+1\right).$$

This process can be repeated only if we can assure that both sides are legitimate. Note that we did not start out in this section with the same example as the previous one. Although we can integrate (wrt s) $e^{-sx}J_0(x)$ to $-e^{-sx}/xJ_0(x)$, the subsequent integral (wrt x) does not converge (in particular, at $x = 0$ we generate $1/0$)!

We chose J_1 for our integration example in this section because the integral on the left converges. (Note that the limit of $J_1(x)/x$ as $x \to 0$ is $1/2$.) In general, confirming the convergence of an integral can require some work (bounding expressions, for example) that is not routine.

4.1.3 Integral transforms

Consider

$$I = \int_L^U f(x)g(x)\ dx$$

and suppose that the limits of the definite integral L and U coincide with those of a known integral transform:

$$\bar{w}(s) = \mathcal{T}[w(x); s] := \int_L^U K(s, x)w(x)\ dx$$

with inverse $\mathcal{T}^{-1}[\bar{w}(s); x]$. Suppose further that K is symmetric in x and s and that both $\mathcal{T}[w; s]$ and $\mathcal{T}^{-1}[w; s]$ exists for all s between L and U. Then

$$I \ = \ \int_L^U \mathcal{T}[f(x); y] \cdot \mathcal{T}^{-1}[g(x); y)\ dy$$

Possible choices for transforms would be Laplace, Fourier-Sine or Fourier-Cosine on $(0, \infty)$, and Fourier (on $-\infty, \infty$)). These each have a symmetric kernel.

These transforms could be used to generate formulas (subject to the imposition of the existence of the transform on $[L, U]$), or in a heuristic venture we have yet to explore, could be used to take a given expression and simplify it prior matching.

4.1.4 Reduction via hypergeometric functions

Adamchik [1] has asserted that many of the entries in the 3-volume encyclopedia [26] can be generated (and were generated by hand simulation) of an algorithm that transforms functions into hypergeometric form. Avgoustis [2] used a similar technique for Laplace transforms of special functions.

These are promising techniques but in practice appear to be error prone and difficult to check; answers in terms of hypergeometric functions are generally less welcome than expressions composed of more elementary functions.

4.1.5 Other techniques

Piquette [25] has written on integration formulas for functions defined through certain kinds of second-order recurrences.

5 Where are we now? Where should we be heading?

This is still a preliminary report on work in progress.

We would like to say that as a result of our production of patterns, major tables of integrals are no longer necessary (or at least are less useful than computer programs.) That is, just as 8-place tables of logarithms and slide-rules have been made obsolete by calculators and computers, large printed tables are not needed. Of course this objective is trivially obtained in some sense, since the tables can be entirely digitized, stored on a magnetic or optical medium, and distributed in the place of physical paper.

We would also need to simulate the useful practice of being able to look at nearby formulas, or perhaps just "browsing." And the tradition of making acknowledgments and corrections for future printing, and

providing references for derivation of the formulas themselves, should be maintained. An instruction section on the use of the computer would be quite different from the current instructions for the use of tables, but would probably also be necessary.

Are we at the point in the development of computer algebra systems so that we have sufficient tools to just sit down and tyoe in the generation algorithm and the patterns? This is unfortunately not the case either. For a variety of reasons, the answer here is no: Systems are generally unable to represent the side-conditions on such formulas. Current system fail in two general ways:

1. They do not support computing with *regions* in the complex plane, or with domains requiring the consideration of several complex variables.

2. *Functions* other than the most straightforward single-valued analytic "expressions" are not representable in a fashion which allows manipulation. Multiple-valued inverses and functions which are piecewise defined (even on the real line) are very difficult to incorporate into the usual operations such as simplification.

These data objects of regions and functions are common in integral tables, and must therefore be represented.

5.0.6 A doubter's view

Some might dismiss much of the special function manipulation, and symbolic integral as a tired "classical" approach to applied mathematics. We imagine such a doubter saying

> These integrals and special functions are merely a handful of the functions which we can now approximate to any desired accuracy by computer-intensive approximations. We can use numeric, semi-numeric (e.g. truncated power series or asymptotic series) tools. To the extent that we have in the past used special functions to understand mathematical physics or other domains, we can now use computation directly to (for example) plot the solutions of differential equations. It may even be faster to use quadrature programs than evaluation some of the complicated "closed form" answers.

Without questioning the usefulness of direct numerical methods, if we are curious as to whether those number or plots are accurate or meaningful we may have to look closer. We can be easily mislead by pretty plots that reflect nothing more than accumulation of round-off error!

Further calculations with numerical data may lead to additional costs and errors of unpredictable severity. Multi-dimensional integration, where symbolic integration of the inside integrals is a standout example of a difficult problem solved in a better way[5]. Furthermore, just as we would be loathe to lose the insight that comes from recognizing the symbolic relationships among sines and cosines, we should recognize that computing with special functions of physics provides insight where approximate series and numerical results would not. We suggest that therefore the success of any effort to replace tables of formulas must recognize the level of insight that should be supported: a serious program to replace tables of integrals and other formulas must include not only symbolic forms when available, but a collection of other tools. For example, these should include executable algorithms for transformations to various forms: symbolic series expansions in power or Laurent series, asymptotic series, and other symbolic-approximate methods. Access to a data-base of relationships and transformations among special functions would be useful in solution of differential equations as well. And naturally, for obtaining numerical results one should be able to produce representations of results that are especially adapted for for numerical computation with high accuracy, and forms especially appropriate for parallel computation (preferably with predictable accuracy too).

Even if we were able to solve all integrals as well as the best table-aided human, there are other considerations that prevent large tables from becoming obsolete.

A table such as Gradshteyn's includes far more heterogeneous information. For example, alternate definitions and notations; closed-form summations; relations among special functions, special values (locations of zeros, singularities, etc.).

[5]If the inner integral has a very complex form, this might not be a major victory over numerical integration, however.

Algorithm based systems depend, for their effectiveness, on "simplification" procedures. Typically the tables provide multiple representations of a given function, and fail to give guidance or heuristics for procedurally transforming expression into a simplified form. As an example, a listing of Gauss' recursion formulas for hypergeometric (${}_2F_1(\alpha, \beta, \gamma, z)$) functions provides 18 relationships among 3 variously "adjacent" functions. Any attempt to make use of these for automatically simplifying a given expression would have to use guidance from other directions.

6 Conclusions

Much remains to be done. The Doubter's points cannot all be refuted, but it is clear that without appropriate representations, we can express neither the questions nor the answers to integral problems and beyond.

7 Acknowledgments

This work was sponsored in part by the National Science Foundation under grant numbers CCR-8812843 and CDA-8722788, the California state MICRO program, IBM Corporation, and by the Defense Advance Research Projects Agency (DoD) under ARPA Order No. 4871, monitored by Space and Naval Warfare Systems Command under Contract No. N00039-84-C-0089.

References

[1] V.S. Adamchik and O.I. Marichev. "The Algorithm for Calculating Integrals of Hypergeometric Type Functions and its Realization in REDUCE System," Proc. ISSAC'90, Tokyo, Japan, Aug., 1990. ACM Press/Addison-Wesley. 212-224.

[2] Yannis Avgoustis. "Symbolic Laplace tranforms of special functions," in *Proc. of the 1977 Macsyma Users' Conf.* NASA CP-2012 Berkeley, CA., July, 1977, 21-41.

[3] Jamil Baddoura, "Integration in Finite Terms and Simplification with Dilogarithms: A Progress Report," *Computers and Mathematics,* E.. Kaltofen & S. Watt (eds.) Springer Verlag, 1989 166-171.

[4] Manuel Bronstein. *Integration of Elementary Functions,* Ph.D. diss. Univ. Calif. Berkeley (Mathematics), 1987.

[5] B. Buchberger, "Should Students Learn Integration Rules?" RISC-LINZ Technical Report 89-07.01 March, 1989 7 pages.

[6] Guy W. Cherry. "Integration in Finite Terms with Special Functions: the Error Function," *J. of Symb. Comp. 1*, Sept, 1985, 283-302.

[7] Guy W. Cherry. "Integration in Finite Terms with Special Functions: the Logarithmic Integral," *J. of Symb. Comp. 2*, 1986, 1-21.

[8] James H. Davenport. *On the integration of algebraic functions,* Springer-Verlag Lect. Notes in Comptr. Sci. 102, Berlin, 1981.

[9] James H. Davenport, Y. Siret, E. Tournier, *Computer Algebra: Systems and Algorithms for Algebraic Computation* Academic Press, 1988.

[10] Adam Dingle. "Branch Cuts in Computer Algebra," MS Report, EECS Dep't, Univ. Calif, Berkeley. 1992.

[11] Richard J. Fateman. "On the Systematic Construction of Algebraic Manipulation Systems," Submitted to *J. Symbolic Computation.*, June, 1987.

[12] Richard J. Fateman, W. Kahan. "Improving Exact Integrals from Symbolic Computation Systems," UC Berkeley - CPAM Technical Report, 1987.

[13] Richard J. Fateman. "Mock Mma" publically available lisp program `mma.src/*`, 1991.

[14] J. Fitch, User-based Integration Software, Proc. 1981 ACM SYMSAC Aug., 1981. 245-248.

[15] K. O. Geddes and T. C. Scott, "Recipes for Classes of Definite Integrals Involving Exponentials and Logarithms," in *Computers and Mathematics*, E. Kaltofen and S. Watt (eds.) Springer-Verlag, 1989, 192-201.

[16] I. S. Gradshteyn and I. M. Ry1zhik, *Table of Integrals, Series, and Products*, Corrected and Enlarged, 1980, 4th edition, 6th printing, Academic Press, 1160 pages + xlv.

[17] Milind Joshi. "Pattern matching for symbolic mathematical transformations," MS. project, Univ. Calif. Berkeley, EECS Dept, 1991.

[18] Peter Klier. "A design for automated integral tables," MS. project, Univ. Calif. Berkeley, EECS Dept, 1989.

[19] Peter Klier and Richard J. Fateman. "On finding the closest bit-wise matches in a fixed set," *ACM Trans. on Math. Softw.* (to appear).

[20] Paul H. Knowles. "Integration of Liouvilliam functions with special functions," ACM-Proc. SYMSAC '86, (1986), 179-184.

[21] Shuichi Moritsugu and Makoto Matsumoto. "A Note on the Numerical Evaluation of Arctangent Functions," SIGSAM Bulletin 23, no. 3.(July, 1989) 8-12.

[22] Joel Moses. "Integration: the Stormy Decade," *Comm. ACM 14* 1971, 548-560.

[23] Joel Moses. "Towards a General Theory of Special Functions *Comm. ACM 15* 1972, 550-554.

[24] Arthur C. Norman. "Integration in Finite Terms." in Buchberger et al. *Computer Algebra: Symbolic and Algebraic Computation*, Springer Verlag, 1983 71-78.

[25] Jean C. Piquette and A. L. van Buren, "Technique for Evaluating Indefinite Integrals Involving Products of Certain Special Functions," *SIAM J. Math. Anal. 15*, no 4, (July, 1984) 845-855.

[26] A. P. Prudnikov, Yu.A. Brychkov, O. I. Marichev. *Integrals and Series,* three volumes, Gordon and Breach Science Publishers, N.Y. 1986-1990.

[27] Robert H. Risch. The problem of integration in finite terms, *Trans. of AMS 139*, 1969. 357-367.

[28] M. Singer, B. Saunders, and B. F. Caviness. "An Extension of Liouville's Theorem on Integration in Finite Terms," *SIAM J. on Computing* vol ? 1985 996-99.0

[29] James Slagle. *A Heuristic program that solves symbolic integration problems in Freshman calculus,* PhD. diss., MIT, 1961.

[30] Harlan Seymour. "Conform, a Conformal Mapping System," in *Proc. of 1986 ACM Symp. on Symbolic and Algebraic Comp.*, B. W. Char (ed). July, 1986, Waterloo, Ontario.

[31] Barry M. Trager. Ph.D. diss., M.I.T. (Elect. Eng. and Comp. Sci.), August, 1985.

[32] Paul S-H. Wang. *Evaluation of Definite Integrals by Symbolic Manipulation.* Ph.D. diss. M.I.T. (Mathematics), August, 1971. also MAC TR 92, Laboratory for Computer Science, 1971.

[33] Stephen Wolfram, *Mathematica: a system for doing mathematics by computer,* Addison-Wesley, 1988, 1990 (2nd edition)