A Knowledge Repository for Indefinite Integration Based on Transformation Rules

A.D. Rich (1) and D.J. Jeffrey (2)

[1] 62-3614 Loli'i Way, Kamuela, Hawaii, USA.
[2] Department of Applied Mathematics
The University of Western Ontario
London, Ontario, Canada N6A 5B7

Abstract. Taking the specific problem domain of indefinite integration, we describe the on-going development of a repository of mathematical knowledge based on transformation rules. It is important that the repository be not confused with a look-up table. The database of transformation rules is at present encoded in Mathematica, but this is only one convenient form of the repository, and it could be readily translated into other formats. The principles upon which the set of rules is compiled is described. One important principle is minimality. The benefits of the approach are illustrated with examples, and with the results of comparisons with other approaches.

1 Introduction

Ever since the automating of the simplification of mathematical expressions was first attempted, there has been a lively discussion as to whether a rule-based approach should complement algorithmic methods, and if so which should be tried first. We contend that a computer algebra system should try rules first, and turn to general purpose algorithms only if no rules apply. This frees developers of algorithms from having to worry about the annoying and trivial problems and the special cases, and instead focus on the genuinely hard and interesting problems.

Unfortunately, rule-based systems, owing to poor implementations, have a reputation for being inefficient and plagued by endless loops. For the problem area of indefinite integration, this paper describes the development of a rule-based repository of knowledge that is compact, efficient, transparent and modular. For the purposes of this preliminary discussion, we shall not address questions of combining our approach with algorithmic approaches, in order to arrive at a full integration system, but concentrate on the questions of constructing a database of knowledge and show examples of how it performs in practice.

It must be emphasized that what is not being described is a scheme for table look-up. Such schemes were described, for example, in [2]. Their approach was to consider data structures and search techniques which would allow them to encode all the entries is reference books such as [1]. Adopting this approach for integration — or *a fortiori* for all simplification — would result in huge databases which would be unwieldy to maintain, debug and utilize. The set of rules whose development is described here is relatively compact, verifiable and efficient.

2 Proper Definition of Transformation Rules

Many of the problems common to rule-based systems can be avoided by defining transformation rules according to the principles below. A transformation rule will be written as $A \to B$, where A and B are mathematical expressions.

– Define Functionally. The right side of a properly defined rule consists of a mathematical expression followed by any required restrictions on the domains of the variables in the rule. Procedural programming constructs such as loops or conditionals are not allowed, and nor are assignments to global or fluid variables. The application of rules defined this way results in a single, comprehensible step in the simplification of an expression.

- Restrict to domains of validity. Many rules are valid only if their variables are restricted to a certain domain. Conditions on a properly defined rule must restrict its application to the domain over which it is valid. For example, the transformation $\sqrt{z^2} \to z$ should only be applied if z is known to be purely imaginary or in the right half of the complex plane (unrestricted versions of this transformation caused the well remembered 'square-root bug' in Maple).
- Restrict to simplification. To avoid infinite loops, applications of rules must eventually result in an expression that can be made no simpler (i.e. an expression to which no rules applies). The conditions attached to a rule must limit its application to those expressions for which its application results in a simpler expression. For example, if F stands for any trigonometric function and n for a rational number, the goal of transformations of $F(n\pi)$ is to reduce the magnitude of the angle. Thus, specifically, although $\sin(n\pi) \to \cos((n - 1/2)\pi)$ is valid for all real and complex n, it should only be applied if n is in the interval $(\pi/2, \pi)$, thereby reducing the angle to the interval $(0, \pi/2)$. Note that if n is negative, application of this rule would actually increase the magnitude of the angle.
- Provision for local variables. Sometimes it is convenient to assign a value to a local variable so it can be used multiple times in a rule's conditions or body without having to recompute it. To provide for this need while preserving the functional nature of rules, assignments to local variables are allowed in properly define rules. However, assignments to fluid and global variables are not allowed.
- Mutually exclusive For a collection of transformation rules to be properly defined, at most one of the rules can be applicable to any given expression. Mutual exclusivity is critical to ensuring that rules can be added, removed or modified without affecting the other rules. Such stand-alone, order-independent rules make it possible to build a rule-based repository of knowledge incrementally and as a collaborative effort.

3 Transformation Rules versus Mathematical Identities

There exist numerous collections of mathematical formulas and identities available in books and on the Internet (e.g. the Wolfram Functions website `functions.wolfram.com` lists over 300,000 formulas). Superficially, properly defined transformation rules and mathematical identities look the same, since both have left and right sides which are mathematical expressions that are equivalent. The obvious question is then why build a repository of knowledge based on rules, when huge libraries of formulas and identities already exist? The answer to that question requires that we understand the difference between collections of rules and collections of identities; they are fundamentally different in nature.

- Rules are active; formulas are passive. Rules are precise instructions on when and how to transform expressions of a particular form into equivalent, but simpler, ones. Identities, on the other hand, are statements of the fact that their left and right sides are mathematically equivalent.
- Rules include application restrictions. Both rules and identities specify the domains of their variables over which they are valid. Properly defined rules include additional conditions so that they are applied only if a simplification actually results and so that collections of rules are mutually exclusive.
- Rule collections are minimized; formula collections are maximized. When crafting a rule-based repository of knowledge, one goal is to minimize the number of rules, by making them mutually exclusive while at the same time maximizing their generality. However, for a library of formulas, one goal is to include all commonly occurring cases of more general formulas, so readers are not required to derive special cases. For example, a library may have dozens of identities giving the algebraic equivalents of trigonometric functions of special angles. However, a repository would have just the handful of rules required to transform trigonometric functions of special angles into algebraic form.
- Integration formulas give final results; Rule collections may not. The ideal entry in an integration table gives an algebraic expression for an integral, and expressing one integral in terms of another is a less satisfactory formula. However, many rules in a repository will express one integral in terms of another, and even if it is possible to express an integral directly in algebraic terms, such a transformation may be excluded in order to keep the repository compact. Thus although ideally an integration table could be used in one pass, a transformation repository will necessarily be recursive.

4 Integration examples

One conspicuous benefit of rule-based integration is the greater simplicity of its results. Simplicity can include not just one integral, but consistent behavior over families of integrals. For example, the following integrals show symmetry between trigonometric and hyperbolic functions:

$$\int \frac{dx}{\sqrt{a+x}\sqrt{b+x}} = 2\,\operatorname{arctanh}\frac{\sqrt{a+x}}{\sqrt{b+x}}\ , \tag{1}$$

$$\int \frac{dx}{\sqrt{a+x}\sqrt{b-x}} = -2\arctan\frac{\sqrt{b-x}}{\sqrt{a+x}}\ . \tag{2}$$

In contrast, Mathematica and Maple express integral (2) using arctangent as shown, but use logarithm for the integral (1):

$$\int \frac{dx}{\sqrt{a+x}\sqrt{b+x}} = \ln\left(a+b+2x+2\sqrt{a+x}\sqrt{b+x}\right)\ . \tag{3}$$

The current version of the repository is being tested on a database of over 5000 integrals, and the results compared with other major computer algebra systems. The comparisons are based on a variety of metrics; for example, simplicity is measured by counting the leaves of the tree structures used to represent the expressions. These metrics, and the results of the comparisons, will be detailed in a future publication.

5 Platform Requirements

An efficient and reliable software platform is required to build a rule-based repository of knowledge. As a minimum the support platform needs to provide the following services:

- Transformation rules. The platform must make it possible to define and recursively apply transformation rules to expressions of a specified form. This requires a flexible and natural syntax for the patterns used to specify the form of expressions.
- Efficient pattern matching. A rule-based system may have thousand of rules, and hundreds of rule applications may be required to simplify an expression. Thus when given an expression to simplify, it is essential that the pattern matcher quickly find the applicable rule, if any. Thoughts on how to implement an efficient pattern matcher is discussed below.
- Exact and arbitrary precision arithmetic. Numerical routines are required for built-in functions and operators since a rule-based approach is usually not appropriate for numerical computations.
- Programming environment. The platform must provide the ability to input, evaluate and display expressions, as well as provide a suitable environment for testing and debugging the repository.
  Since most modern computer algebra systems provide the above capabilities, they are suitable platforms for crafting a rule-based repository of knowledge.

6 Efficient Pattern Matching

A general purpose repository might require thousands or even tens of thousands of rules. Obviously sequentially searching a list of that many rules to find a match would be unacceptably slow. Even having a separate list of rules for each built-in function or operator is insufficient, since some functions may have a large number of rules associated with it (e.g. our integrator requires over a 1000 rules).

Therefore instead of a list, the software platform supporting a repository should store the rules in the form of a discrimination net based on the tree structure of expressions. Then, for example, all rules applicable to expressions of the form sin(u) will be collected in one branch of the tree, all differentiation rules in another, all integration rules in another, etc. Then the rules in each branch will be recursively subdivided based on the form of its arguments, etc.

With the rules stored in such a discrimination net, the rule applicable to a given expression can be quickly found by a simple tree walk in $\log(n)$ time, where n is the number of rules in the repository.

7 Advantages

The following summarizes the advantages of storing mathematical knowledge in the form of a repository based on properly defined transformation rules:

- Human and machine readable. Since rules are defined using mathematical formulas rather than procedural programming constructs, they express a self-contained mathematical fact that can be attractively displayed in standard two-dimensional mathematical notation.
- Able to show simplification steps. The successive application of rules exactly corresponds to the steps required to simplify an expression. Thus when a rule is applied, it can display itself in standard mathematical notation as justification for the step, and then suspend further simplification so the partially simplified result is returned.
- Mechanical rule verification. Since the right side of a properly defined rule is just a mathematical expression, the rules validity can often be mechanically verified. For example, the right side of integration rules can be differentiated to see if they equal the integrand on the left.
- Facilitates program development. The fact that properly defined rules are inherently self-contained and free of side-effects makes it easy to test the effect on the system of selectively adding, modifying or deleting rules. Although collections of rules may be highly recursive, each individual rule must be able to stand on its own, thus making it possible to test it on examples before adding it to the collection.
- Platform independent. Since properly defined transformation rules consists only of mathematical expressions and pattern matching specifications, the translation of rules from the syntax of one computer algebra system to another is relatively straight-forward.
- White box transparency. For the most part, computer algebra systems appear as mysterious black boxes to users with little or no explanation given as to how results are obtained. However, if the source file of rules on which a CAS is based were included with the system, it becomes a transparent white box, making it possible for users to modify existing rules and even add new ones.
- Fosters community development. The open source nature of a rule-based repository of knowledge would foster an active community of users. A website blog dedicated to a repository could provide developers the ability to propose new rules and improvements to existing ones. Developers would vie with one another to get credit for adding new rules to the repository. Others would shoot down defective ones. Thus the system would grow and evolve in Darwinian fashion much the same way Wikipedia does.
- An active repository. Encyclopedias and reference manuals, even on-line ones, are inherently passive repositories in the sense that users have to find the knowledge required to solve a given problem, and then manually apply it. However, given a problem a rule-based repository actively finds and applies the knowledge required to solve it. Thus the knowledge in such repositories is in a much more useful form.

References

1. Abramowitz,M. & Stegun,I., *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. US Government Printing Office, 1964. 10th Printing December 1972.
2. Einwohner, T.H. & Fateman, Richard J., *Searching techniques for integral tables*, Proceedings ISSAC '95, pp 133-139, ACM Press, 1995.