Definite Integration of Parametric Rational Functions: Applying a DITLU

A. A. Adams[*]
School of Computer Science,
University of St Andrews
aaa@cs.st-and.ac.uk

November 2, 2003

Abstract. *In [2] we presented a Definite Integral Table Lookup (the DITLU) for parametric functions, including a minimal prototype implementation demonstrating its capabilities. In this paper we present a possible application of a DITLU, which would extend its utility for a modest investment of effort. The naive algorithm for indefinite integration of rational functions (see e.g. [12, §2.10]) can be implemented for parametric rational functions. This involves splitting the rational function integrand using partial fractions. The resulting integrands all fall within a limited class which may be covered in a DITLU by a very small number of table entries. Extensions of this idea to less naive integration algorithms, and the number of table entries required to implement them, are also considered.*

1 Introduction

Definite integration is acknowledged [15] to be a tricky problem for Computer Algebra Systems [CAS] when a symbolic answer is required or when the original query involves parameters, i.e. in those cases where the well-developed numeric methods are inapplicable. In [2] and elsewhere we presented a Definite Integral Table Lookup (the DITLU) for calculating definite integrals involving parameters in the integrand and the limits. We implemented a prototype version of the table

[*] This work is supported by the UK EPSRC: grant GR/L48256.

to perform proof of concept, and described the extensions necessary for a production version of the system to be included in a CAS. The DITLU uses automated theorem proving (ATP) technology to winnow inconsistent cases from the multiplicity that occur in examples with a number of parameters. Our previous papers ([1, 2, 3]) also described how the contents of the DITLU might be formally verified correct using machine assisted theorem proving (the difficulty of the proofs is beyond current automatic techniques). While finalising the form of the prototype it occurred to us that some simple extensions of the table would allow it to solve far more cases than those actually included in the table. In this paper we present an example of this: the extensions to the system and the table entries required for the DITLU to be instrumental in the correct calculation of any definite integral of a parametric rational function.

First we begin with a review of our DITLU system, in section 2. Then in section 3 we present the naive algorithm for integrating rational functions with parameters using the DITLU. In section 4 we consider other indefinite integration algorithms and how their benefits could be included in our methods. Our conclusions are presented in section 5. In the appendix 1 we include the table entries that would need to be added to the DITLU for the naive algorithm to be used.

2 Review of the DITLU

Definite integration in the presence of parameters is a tricky problem, and one for which CAS often return incorrect, or partially correct, answers. Much of this problem is due to design issues, mainly historical, which have produced programs very good at performing algebraic calculation, but poor at analytic calculation. One of the primary problems for CAS designers when attempting to answer analytic queries is that analysis is much concerned with ranges of definition, points of undefinedness and actual values of functions, whereas algebra is a much more abstract topic, in which any function satisfying $D(f) = f$, where D is a differential operator, is regarded as *an* exponential function. Analysis, on the other hand is concerned with the difference, as much as the similarities, between e^x and $e^{(x+1)}$. While CAS are designed to calculate algebraic quantities they are not good at keeping track of the logical side conditions of analysis. There have been many attempts to develop mathematical software which retains a high level of logical consistency, but mostly these have ended up quite well separated from the mainstream user of mathematical software such as Maple or Mathematica. Our approach was to consider the shortcomings of existing CAS and consider how ATP technology could be brought to bear upon these problems. One result of our efforts is the DITLU. The DITLU consists of a table of parametric definite integrals. A user may submit a query to the table with complicated constraints on the side conditions. It is our opinion that useful parametric queries are often highly constrained, and that the inability of CAS to cope well with the analytic ramifications of such constraints is one of their weaknesses. For example, a user may be interested in:

$$\int_{0}^{b} \frac{1}{x-1}\, dx \qquad (b > 0) \wedge (b \neq 1) \tag{1}$$

while allowing the use of the Cauchy Principal Value method, which allows positive and negative areas on either side of a pole to cancel each other out. The CAS **axiom** has no ability to perform definite integration through poles using the Cauchy Principal Value method. Design flaws and the inability to cope well with side conditions on parameters lead Maple, Mathematica and Matlab to all return $\ln(b-1) - i\pi$, which is correct for $b < 1$, albeit only via complex arithmetic, but incorrect for $b > 1$, giving a complex answer. The prototype DITLU can produce the correct answer to this query, since it contains the table entry shown in figure 1.

To accommodate the limitations of pattern matching in the prototype we submit our query thus:

$$\int_{0}^{k} \frac{1}{1.x-1}\, dx \qquad (k > 0) \wedge (k \neq 1) \tag{2}$$

and the system then matches it against the entry shown in figure 1, with table variables matched thus:

$$q \rightarrow 1; \qquad r \rightarrow -1; \qquad b \rightarrow 0; \qquad c \rightarrow k \tag{3}$$

The prototype DITLU then calls the theorem prover PVS to attempt to prove eight conjectures, corresponding to the eight parts of the answer in figure 1. Each conjecture represents an attempt to prove that a part of the answer is unnecessary for the query submitted because the constraints on the query cannot be satisfied together with the constraints on that part of the answer. Here is an example of a conjecture that is proved by PVS, allowing the system to remove the first line in figure 1 from the answer returned:

$$\neg\exists k\!:\!\mathbb{R}.(k = 0) \wedge ((k > 0) \wedge (k \neq 1)) \tag{4}$$

The first part of this ($k = 0$) is the constraints from the table entry ($b = c$), dereferenced according to the variable assignments in (3), the rest are the query constraints. Similarly the table will be able to prove that lines 1, 2, 4, 6, 7 and

$$\int_b^c \frac{1}{qx+r}\,dx \;=$$

	Result	*Constraints*
1	0	$(b=c)$
2	Undefined	$(b \neq c) \wedge (q \neq 0) \wedge \left(\left(b = \frac{-r}{q}\right) \vee \left(c = \frac{-r}{q}\right)\right)$
3	$\ln\lvert qc+r\rvert - \ln\lvert qb+r\rvert$	$(b \neq c) \wedge (q \neq 0) \wedge \left(b < \frac{-r}{q}\right) \wedge \left(c < \frac{-r}{q}\right)$
4	$\ln\lvert qc+r\rvert - \ln\lvert qb+r\rvert$	$(b \neq c) \wedge (q \neq 0) \wedge \left(b > \frac{-r}{q}\right) \wedge \left(c > \frac{-r}{q}\right)$
5	$\ln\lvert qc+r\rvert - \ln\lvert qb+r\rvert$ (CPV)	$(b \neq c) \wedge (q \neq 0) \wedge \left(b < \frac{-r}{q}\right) \wedge \left(c > \frac{-r}{q}\right)$
6	$\ln\lvert qc+r\rvert - \ln\lvert qb+r\rvert$ (CPV)	$(b \neq c) \wedge (q \neq 0) \wedge \left(b > \frac{-r}{q}\right) \wedge \left(c < \frac{-r}{q}\right)$
7	$\frac{c-b}{r}$	$(b \neq c) \wedge (r \neq 0) \wedge (q = 0)$
8	Undefined	$(b \neq c) \wedge (r = 0) \wedge (q = 0)$

Figure 1. A DITLU Entry

8 have unsatisfiable constraints for this query and remove them. The other lines give rise to false, and therefore unprovable, conjectures such as:

$$\neg\exists k : \mathbb{R}.((0 \neq k) \wedge (1 \neq 0) \wedge (0 < 1) \wedge (k > 1)) \wedge \\ ((k > 0) \wedge (k \neq 1)) \tag{5}$$

So the answer the DITLU returns is:

$$\begin{array}{c|c}
\cline{2-2}
\ln|1 \times k - 1| - \ln|1 \times 0 - 1| & \begin{array}{c}(0 \neq 1) \wedge (1 \neq 0) \wedge \\ (0 < 1) \wedge (k < 1)\end{array} \quad (6) \\
\cline{2-2}
\begin{array}{c}\ln|1 \times k - 1| - \ln|1 \times 0 - 1| \\ (\mathrm{CPV})\end{array} & \begin{array}{c}(0 \neq k) \wedge (1 \neq 0) \wedge \\ (0 < 1) \wedge (k > 1)\end{array} \quad (7) \\
\cline{2-2}
\end{array}$$

One might ask why a theorem prover is used for this constraint checking, rather than a decision procedure for real arithmetic or a constraint solver. The answer to this is that at present, the prototype implementation may indeed be no more successful than these techniques. However, the use of a theorem prover provides scope for extensions impossible with other approaches. For instance, the prototype implementation makes use of the equational aspects of the deduction engine to support the matching algorithm, in a similar way to the MFD2 system [6]. Existing constraint satisfaction systems all work with floating point representations and therefore use, for instance, approximations to irrational numbers such as π. Decision procedures for the real numbers are only available for very limited problem sets and extension of the constraints to include transcendental functions of the parameters quickly puts the constraint satisfaction problem outside the application are for such systems. We are currently exploring ideas for solving such problems automatically with PVS, using ideas such as those proposed by Sterling et al in [14].

2.1 Details of the Prototype

The prototype implementation is written in Allgero Common Lisp and comprises approximately 5000 lines of code. We have also extended the existing PVS real analysis library by Dutertre [8] to include some transcendental functions ($\exp$, $\log$ and the trig and inverse trig functions) [11]. We are currently considering the existing automation available in PVS and its utility in proving constraints involving transcendental functions.

The interface between PVS and the prototype is crude, consisting of output printed by the lisp session which may then be copied and pasted into a PVS session. Since PVS is also implemented in Allegro Common Lisp, we are investigating ways to link the two systems more closely.

3 Naive Algorithm for Integrating Rational Functions

In this section we present the method for using a small number of simple integrals as DITLU table entries to calculate definite integrals for all rational functions of one variable over the reals. The algorithm is based on the naive method for calculating indefinite integrals for rational functions.

We are considering quotients of polynomials which take real values of a single variable. We are only considering parameters which appear in the coefficients of the polynomials, not in the exponents. In order to use the DITLU to integrate such functions it might appear that an infinite number of entries would be required. In fact even allowing the coefficients of the polynomials to contain transcendental functions in the parameters, all rational functions can be decomposed to use one of four DITLU entries. Much pre-processing of the original rational function integrand is required however, and the complexity of the pre-processing appears to be very high (see equation (8) for an example of the pre-processing required). Nevertheless we feel this is an interesting possible application/extension of the DITLU.

The following algorithm is described in [12, $\S$2.10] for indefinite integration, and the table entries are developed from the indefinite integrals shown in that section.

From an original query of:

$$\int_t^u \frac{P_1(x)}{P_2(x)}\, dx, C$$

where C is a set of constraints on the parameters occurring in t, u, P_1 and P_2, we first reduce the integral to a sum of integrals of the form:

$$\int_t^u P_3(x)\, dx + \int_t^u \frac{P_4(x)}{P_2(x)}\, dx, C'$$

where $\deg(P_4) < \deg(P_2)$. The first half of the sum is always a well-defined integral that current algorithms can deal with satisfactorily. Errors in the output from CAS implementations of these algorithms are caused by bugs elsewhere within the systems and not our concern. So, we will focus on the problem of integrating rational functions where the degree of the denominator is strictly greater than the degree of the numerator. The main problem with definite integration derives from discontinuities in the integrand. For the rational functions such discontinuities only occur at easily defined places: at the roots of the denominator polynomial. So, we must calculate the roots of P_2. This is a computationally hard problem in the parametric case, but not a difficult algorithm. ATP technology might well be of use here in ruling out cases with inconsistent constraints on the values of the parameters. P_2 may be decomposed thus:

$$P_2 = Q_1^{n_1} \ldots Q_k^{n_k} t^{m_1} \ldots L_j^{m_j}$$

where the Q_i are all quadratics with no real roots and the L_i are linear and hence have exactly one real root. For example, we may decompose:

$$x^3 + cx$$

to

$$\begin{array}{ll} x(x+\sqrt{-c})(x-\sqrt{-c}) & \text{if } c < 0 \\ (x)^3 & \text{if } c = 0 \\ x(x^2+c) & \text{if } c > 0 \end{array}$$

From the factorisation of P_2 we may use partial fractions to decompose:

$$\frac{P_4}{P_2} = \sum_{i=1}^{k} \left(\frac{R_i}{Q_i^{n_i}} \right) + \sum_{i=1}^{j} \left(\frac{S_i}{L_i^{m_i}} \right)$$

where $\deg(R_i) < 2n_i$ and $\deg(S_i) < m_i$. If $\deg(R_i) > 1$ or $\deg(S_i) > 0$, we may decompose them thus:

$$\frac{R}{Q^n} = \frac{a_1x+b_1}{Q} + \frac{a_2x+b_2}{Q^2} + \ldots + \frac{a_nx+b_n}{Q^n}$$

$$\frac{S}{L^m} = \frac{c_1}{L} + \frac{c_2}{L^2} + \ldots + \frac{c_m}{L^m}$$

A more concrete example illustrates the factorisation more clearly:

$$\frac{x}{(x+1)^3} = \frac{0}{(x+1)} + \frac{1}{(x+1)^2} + \frac{(-1)}{(x+1)^3}$$

So, if we are calculating:

$$\int_t^u \frac{1}{x^3+cx}\, dx \tag{8}$$

we decompose this into its factors and partial fractions thus:

$$\begin{array}{ll} \frac{1}{c}\int_t^u \frac{1}{(x)^1}\,dx - \frac{1}{2c}\int_t^u \frac{1}{(x+\sqrt{-c})^1}\,dx - \frac{1}{2c}\int_t^u \frac{1}{(x-\sqrt{-c})^1}\,dx & \text{if } c < 0 \\ \int_t^u \frac{1}{(x)^3}\,dx & \text{if } c = 0 \\ \frac{1}{c}\int_t^u \frac{1}{(x)^1}\,dx - \frac{1}{c}\int_t^u \frac{x}{(x^2+c)^1}\,dx & \text{if } c > 0 \end{array}$$

Note that extraneous constant factors have been pushed outside the integrals.

So, if we had DITLU entries covering the following integrals, we can deal with all rational integrals in full safety.

$$\begin{array}{ll} \int_t^u \frac{m}{(x+c)^i}\,dx & i \in \mathbf{Z}^+ \\ \int_t^u \frac{mx+n}{(x^2+bx+c)^i}\,dx & i \in \mathbf{Z}^+ \wedge b^2 < 4c \end{array}$$

The general form of these integrals (see figures 2 and 3) as produced from the indefinite integrals [12, §2.103,{1–5}] by applying the Fundamental Theorem of Calculus, includes recursive entries (both entries for “quadratic” queries are recursive). Our prototype does not include a method of entering recursive integrals into the table, but there is no apparent reason why such a system could not be included in a production version of the DITLU. Note that answers for these entries which would not be necessary for this algorithm have been entered as “Unknown” for the sake of brevity.

We must note that there is a high computational complexity associated with the factorisation step involved in this algorithm. Full factorisation of a parametric rational function is inherently an expensive operation. The ATP technology used to support the DITLU should be equally as useful, however, in supporting this parametric factorisation, by helping to identify unsatisfiable sets of constraints on branches of the factorisation. It is our contention that any parametric query that is useful will be highly constrained. It also seems reasonable that highly constrained queries will factorise without too many case splits on parameter values (randomly this may not be the case but for problems of interest we feel this is a reasonable assumption). Provided we have a factorisation algorithm that copes with constraints on the values of the parameters, we feel that the naive algorithm is a reasonable approach in itself. This naive algorithm is not used in CAS because of its high computational complexity and also because of the introduction of unnecessary algebraic numbers into the answer. We therefore consider the possible utility of using the DITLU in a similar fashion with the more advanced algorithms used in computing indefinite integrals of rational functions.

4 Extensions to the Algorithm

In this section we consider the common variant methods used in calculating indefinite integrals of rational functions, their application to the problem of definite integration, and how this affects our proposal. We begin with a brief discussion of other methods in subsection 4.1, and then discuss whether those gains carry over to the definite integration case in subsection 4.2. We discuss in section 5 how we might change our algorithm above to make use of the efficiency gains of these other methods while retaining confidence in the answer.

4.1 Other Methods

We begin with a parametric definite integral of a rational function:

$$\int_a^b \frac{N}{Q}\, dx \tag{9}$$

As mentioned above, we are only interested in cases where $\deg(N) < \deg(Q)$, since the definite integration of polynomials themselves is a very simple calculation requiring only care in the programming to be robust and complete, even in the presence of parameters.

The Ostrogradskiy-Hermite Method

The Ostrogradskiy-Hermite Method [12, §2.104] (also called simply Hermite's Method in [10, §11.3]) simplifies the indefinite integral of a rational function thus:

$$\int \frac{N}{Q}\, dx = \frac{P}{R} + \int \frac{L}{M}\, dx$$

where $\deg L < \deg M$ and M is monic and square-free. The rational function $\frac{P}{R}$ is referred to as the rational part of the integral. The indefinite integral remaining is called the logarithmic part of the integral, because it requires answers involving logarithmic extensions.

The core of Hermite's method is a square-free factorisation of the original denominator polynomial, combined with some use of integration by parts. Therefore, using Hermite's method on a rational function involving parameters would appear to be no more difficult than factoring the original parametric polynomial Q. However, since we are concerned with definite integrals we must remember:

$$\int_a^b \frac{N}{Q}\,dx = \left[\frac{P}{R}\right]_a^b + \int_a^b \frac{L}{M}\,dx$$

so that care must be taken in substituting the limits into the rational part. In fact, this substitution is as tricky as any other definite integration of a rational function in that we must be aware of any roots of R in $[a, b]$.

Considering this limit substitution we realise that the solution to this problem requires only a simple extension of the DITLU. An identical matching algorithm and constraint checking system can be applied to the problem of limit substitution, so all we need to apply the DITLU to this problem is to limit substitution is a flag distinguishing between integrals and limit substitutions. Two tables, one of integrals and one of limit substitutions could then use the same code.

Horowitz' Method

Horowitz' method is a variation of Hermite's method for calculating the rational part of the indefinite integral. Horowitz' method would appear no more difficult to apply in the parametric case than Hermite's method, and so for our purposes the result is the same.

The Rothstein-Trager Method

We now turn our attention to the methods for calculating the logarithmic part of the integral. The aim of this method, and others such as the Lazard-Rioboo-Trager improvement, is to avoid performing calculations in any larger an extension field than is absolutely necessary. It is fairly common for the naive method described in section 3 to produce answers involving spurious algebraic extensions such as $\sqrt{a}$, $\sqrt{a\sqrt{b}}$ etc. which may be simplified out into expressions simply involving a and b. For example:

$$p > 0 \qquad \frac{x^3}{x^4+p^2} = \frac{x^3}{(x^2-\sqrt{2p}x+p)(x^2+\sqrt{2p}x+p)}$$

and so the answer provided by the naive method would involve $\sqrt{2p}$, whereas since

$$p > 0 \Rightarrow x^4+p^2 > 0 \text{ and } \frac{\mathrm{d}}{\mathrm{dx}}(x^4+p^2) = 4x^3$$

the simple result of a definite integral is:

$$p > 0 \qquad \int_a^b \frac{4x^3}{x^4+p^2}\,dx = \ln(b^4+p^2) - \ln(a^4+p^2)$$

which does not involve algebraic extensions.

The extension of Rothstein-Trager and its improvements to include parametric rational functions is not something that has been done in any existing CAS, although Bronstein's group are currently working on implementing various parametric algorithms properly, as opposed to the ad hoc implementations currently in use, as part of their $\Sigma^I T$ library [4].

4.2 Discussion of Other methods

In the absence of implementations of the more advanced methods such as Rothstein-Trager and Horowitz for parametric rational functions, it is difficult to judge how much benefit they bring to the problem in this case. Such methods were at least partly developed because computation in algebraic extension fields was expensive. This is almost certainly no longer the case. There is also something to be said for implementing simple algorithms when one is at least as interested in the validity of the output as in the time taken to compute that output.

5 Conclusions

In order to use the above methods instead of a full factorisation one would need to extend the DITLU in two ways. More table entries would be needed than those shown in appendix 1. Our original algorithm required only four table entries (fewer with a more sophisticated DITLU than our prototype) to cover any rational function. We need only consider one table entry for each degree of the denominator, since

$$\frac{a_n x^n + \cdots + a_1 x + a_0}{b_{n+1}x^{n+1} + \ldots + b_1 x + b_0}$$

matches

$$\frac{c_k x^k + \cdots + c_1 x + c_0}{d_{n+1}x^{n+1} + \ldots + d_1 x + d_0}$$

for any $k \leq n$, with $a_i = 0$ for $i > k$.

We may assume that for any well-constrained problem there will be a reasonable level of factorisation, and so entries up to denominators of degree n will be sufficient to allow full answers to be calculated for a large number of queries up to degree n^2, and a decreasing number of queries of higher degree.

This leaves the problem of computing the limit substitution for the “rational part” of the indefinite integral. At first this may appear a relatively difficult problem but it is also soluble using a variant of a DITLU. There is almost no difference between table entries comprising integrals of rational functions and limit substitutions into rational functions. Only the table compilation is different. Table operation is identical provided we add a distinguishing marker to differentiate between integrals and limit substitutions.

So, whether using the simple, yet computationally expensive, naive method, or the more complicated and more efficient methods, the problems of using the indefinite integral to compute a definite integral may be bypassed by astute use of a DITLU. A limited number of entries may be used for computing a much larger number of queries, including those of higher degree.

References

[1] 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. http://www-theory.cs.st-and.ac.uk/publications/CAAR/CS993.

[2] A. A. Adams, H. Gottliebsen, S. A. Linton, and U. Martin. Automated theorem proving in support of computer algebra: symbolic definite integration as a case study. In Dooley [7], pages 253–260.

[3] A. A. Adams, H. Gottliebsen, S. A. Linton, and U. Martin. VSDITLU: a verifiable symbolic definite integral table look-up. In Ganzinger [9], pages 112–126.

[4] M. Bronstein. SUM-IT: A strongly-typed embeddable computer algebra library. In Calmet and Limongelli [5].

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

[6] S. Dalmas, M. Gaëtano, and C. Huchet. A Deductive Database for Mathematical Formulas. In Calmet and Limongelli [5].

[7] S. Dooley, editor. *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation*. ACM Press, 1999.

[8] B. Dutertre. Elements of Mathematical Analysis in PVS. In von Wright et al. [16], pages 141–156.

[9] H. Ganzinger, editor. *Automated Deduction — CADE-16*. Springer-Verlag LNAI 1632, 1999.

[10] K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer, 1992.

[11] H. Gottliebsen. Transcendental Functions and Continuity Checking in PVS. In Harrison and Aagaard [13], pages 198–215.

[12] I. S. Gradshteyn and I. M. Ryzhik. *Table of Integrals, Series and Products*. Academic Press, 1965.

[13] J. Harrison and M. Aagaard, editors. *Theorem Proving in Higher Order Logics: 13th International Conference, TPHOLs 2000*. Springer-Verlag LNAI 1869, 2000.

[14] L. Sterling, A. Bundy, L. Byrd, R. O'Keefe, and B. Silver. Solving symbolic equations with PRESS. *J. Symbolic Comput.*, 7(1):71–84, 1989.

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

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

1 Table Entries for the Naive Algorithm

$$i \in \mathbf{Z} \quad \int_t^u \frac{dx}{(x+c)^{2i}} =$$

Result	*Constraints*
0	$t = u$
unknown	$(t \neq u) \wedge (i < 1)$
undefined	$(t \neq u) \wedge (i > 0) \wedge$ $((t = -c) \vee (u = -c))$
undefined	$(t \neq u) \wedge (i > 0) \wedge$ $(((t < -c) \wedge (u > -c)) \vee$ $((t > -c) \wedge (u < -c)))$
$\frac{1}{(2i-1)}\left(\frac{1}{(t+c)^{2i-1}} - \frac{1}{(u+c)^{2i-1}}\right)$	$(t \neq u) \wedge (i > 0) \wedge$ $(((t < -c) \wedge (u < -c)) \vee$ $((t > -c) \wedge (u > -c)))$

$$i \in \mathbf{Z} \quad \int_t^u \frac{dx}{(x+c)^{(2i+1)}} =$$

0	$t = u$
unknown	$(t \neq u) \wedge (i < 0)$
undefined	$(t \neq u) \wedge (i \geq 0) \wedge$ $((t = -c) \vee (u = -c))$
$\ln\left(\frac{\lvert u+c \rvert}{\lvert t+c \rvert}\right)$	$(t \neq u) \wedge (i = 0) \wedge$ $(t \neq -c) \wedge (u \neq -c)$
$\frac{1}{2i}\left(\frac{1}{(t+c)^{2i}} - \frac{1}{(u+c)^{2i}}\right)$	$(t \neq u) \wedge (i > 0) \wedge$ $(t \neq -c) \wedge (u \neq -c)$

Figure 2. Entries for "Linear" Queries

$$i \in \mathbf{Z} \quad \int_t^u \frac{(mx+n)}{(x^2+bx+c)^i}\, dx =$$

0	$t = u$
unknown	$(t \neq u) \wedge$ $((i < 1) \vee (b^2 \geq 4c))$
$\frac{m}{2} \ln\left(\frac{(u^2+bu+c)}{(t^2+bt+c)}\right) + \frac{(2n-mb)}{\sqrt{4c-b^2}}\left(\tan^{-1}\left(\frac{(2u+b)}{\sqrt{4c-b^2}}\right) - \tan^{-1}\left(\frac{(2t+b)}{\sqrt{4c-b^2}}\right)\right)$	$(t \neq u) \wedge$ $(i = 1) \wedge (b^2 < 4c)$
$\frac{nb-2mc+(2n-mb)u}{(i-1)(4c-b^2)(u^2+bu+c)} - \frac{nb-2mc+(2n-mb)t}{(i-1)(4c-b^2)(t^2+bt+c)} + \frac{(2i-3)(2n-mb)}{(i-1)(4c-b^2)} \int_t^u \frac{1}{(x^2+bx+c)^{i-1}}\, dx$	$(t \neq u) \wedge$ $(i > 1) \wedge (b^2 < 4c)$

$$i \in \mathbf{Z} \quad \int_t^u \frac{1}{(x^2+bx+c)^i}\, dx =$$

0	$t = u$
unknown	$(t \neq u) \wedge$ $((i < 1) \vee (b^2 \geq 4c))$
$\frac{2}{\sqrt{4c-b^2}}\left(\tan^{-1}\left(\frac{(2u+b)}{\sqrt{4c-b^2}}\right) - \tan^{-1}\left(\frac{(2t+b)}{\sqrt{4c-b^2}}\right)\right)$	$(t \neq u) \wedge$ $(i = 1) \wedge (b^2 < 4c)$
$\frac{(b+2u)}{(i-1)(4c-b^2)(u^2+bu+c)} - \frac{(b+2t)}{(i-1)(4c-b^2)(t^2+bt+c)} + \frac{(4i-6)}{(i-1)(4c-b^2)} \int_t^u \frac{1}{(x^2+bx+c)^{i-1}}\, dx$	$(t \neq u) \wedge$ $(i > 1) \wedge (b^2 < 4c)$

Figure 3. Entries for “Quadratic” Queries