Mathematical Knowledge Representation (Extended Abstract)

James H. Davenport*

Department of Computer Science, University of Bath, Bath BA2 7AY, England
J.H.Davenport@bath.ac.uk

Abstract. Mathematical knowledge manipulation requires mathematical knowledge representation. This is not as easy as it seems, and there are various ambiguities in mathematics as commonly described. This paper describes some of these, and, where relevant, outlines how the OpenMath Content Dictionary mechanism can handle them.

1 Introduction

Before we can talk about the "management" of mathematical knowledge, we have to talk about its representation, and the problems of doing this. Here we discuss some of the pitfalls that the author has encountered, ranging from the concrete to the more philosophical (and fundamental). These are generally described from an OpenMath [17] point of view, but the majority of the problems are inherent to the representation of mathematical knowledge, rather than accidents of OpenMath.

2 Exactly which ... do you mean?

Mathematical notation is not quite as unambiguous as we would like (and as mathematicians like to pretend). This is a problem that pervades much of mathematics, and we give just a couple of examples.

Branch cuts Whenever a function f from $\mathbf{C}$ to $\mathbf{C}$ (or $\mathbf{R} \to \mathbf{R}$) is not injective, the inverse needs to be defined on only a subset of $\mathbf{C}$ (or $\mathbf{R}$), typically defined by taking some injective subset of f, i.e. f defined on some subset of $\mathbf{C}$ (or $\mathbf{R}$) by some "branch cuts". There is no fully logical way to choosing these branch cuts, and some standard decision has to be taken, e.g. as in [1]. These decisions are not always clear, and many alternative choices can be justified: e.g. it is common these days to have $-\pi < \Im \log z \leq \pi$, but the author was originally taught the range $0 \leq \Im \log z < 2\pi$.
For the inverse trigonometric and hyperbolic functions, this issue is discussed in [5]. They quote the amusing example of arccot in table 1.

The OpenMath solution to this problem is to be very explicit in the Content Dictionaries for these functions, defining them all in terms of the branch cut for log, so that arccot, as the symbol

* Partially supported by the E.U. OpenMath Thematic Network

Table 1. value of $\operatorname{arccot}(-1)$ in various sources

[1]	1st printing	$3\pi/4$	inconsistent
[1]	9th printing	$-\pi/4$	
[11]	5th edition	?	inconsistent
[20]	30th edition	$3\pi/4$	inconsistent
Maple	V release 5	$3\pi/4$	
Axiom	2.1	$3\pi/4$	
Mathematica	[18]	$-\pi/4$	
Reduce	3.4.1	$-\pi/4$	in floating point
Matlab	5.3.0	$-\pi/4$	in floating point
Matlab	5.3.0	$3\pi/4$	symbolic toolbox

```
<OMS name="arccot" cd="transc1">
```

is defined as $\operatorname{arccot}(z) = \frac{1}{2i}\ln\left(\frac{z+i}{z-i}\right)$.
There are other differences: for example Derive and Maple differ in their definition of inverse tangents, so that

$$\underbrace{\arctan}_{\text{Derive}}(z) = \overline{\underbrace{\arctan}_{\text{Maple}}(\overline{z})}.$$

Here the difference is only on the branch cuts, a set of measure zero, so the difference is less likely to be noticed in testing.
Of course, it would be possible to have other CDs, and the author is contemplating writing one for the multivalued versions[1] of these, so that

```
<OMS name="Arccot" cd="transc3">
```

would be defined as $\operatorname{Arccot}(z) = \{w \mid \cot(w) = z\}$.

Algebraic Structures Most algebraic structures have names, and we are generally happy working with these — we all know what $\mathrm{PSL}(3, \mathbf{Z})$ is (though some of us might write it as $\mathrm{PSL}_3(\mathbf{Z})$, which explains some of the difficulty of translating a presentation-oriented language such as LaTeX or MathML–presentation [19] into a content-oriented language such as OpenMath [17]). It is the OpenMath view that there is one abstract mathematical object, say

```
<OMA>
  <OMS name="PSL" cd="lineargroups"/>
  <OMI> 3</OMI>
  <OMS name="Z" cd="setname1"/>
</OMA>
```

and that whether it is rendered as $\mathrm{PSL}(3, \mathbf{Z})$ or $\mathrm{PSL}_3(\mathbf{Z})$ is up to the configuration of the browser, just as the rendering of

[1] The reasons why these are not the fundamental OpenMath functions are given in [6].

```
<OMA>
  <OMS name="interval_oc" cd="interval1"/>
  <OMI> 3</OMI>
  <OMI> 4</OMI>
</OMA>
```

as $(3,4]$ or $]3,4]$ depends on whether the browser has been configured to produce English or French notation.

However, there are still several ambiguities. We all accept that C_6 acts on six elements, and has size six, but what of D_6? Does it act on six elements (and therefore have size 12) or have size 6 (and therefore act on three elements, and be isomorphic to S_3)? The situation is described in [3] as follows.

> A name of the form $\langle\langle X(n)\rangle\rangle$ denotes the n-th member of a family as a permutation group on n points. The same group is denoted by $\langle\langle X_n\rangle\rangle$ if it occurs as an abstract group, but not necessarily with this action. Exceptions are dihedral and Frobenius groups, for which traditionally the *size* is indicated by an index. Thus $\langle\langle D(4)\rangle\rangle = \langle\langle D_8(4)\rangle\rangle$ is the dihedral group of size 8.

However, this convention is far from universal. Furthermore, the notational scheme described in [3] (itself based on that in [4]) can lead (inevitably) to several names for the same group, e.g. $F_{18}(6) = [3^2]2 = 3 \wr 2$ or $E(8) : A4 = [\frac{1}{3}A(4)^2]2 = e(4) : 6$.

Again, the remedy would be that of being very clear in the CDs describing these objects, with Formal Mathematical Properties to express (at least some of) the alternative names for groups.

3 Just how constructive are we?

For the moment, let us assume that an integral domain is a well-known concept (though we will see later that there are problems with this). then we can define two concepts specialising this.

GCD domain An integral domain R is a GCD domain if, and only if, for any two elements a and b of R, there is an element g of R such that:
1. $g|a$ and $g|b$;
2. if $c|a$ and $c|b$, then $c|g$.

Unique factorisation domain An integral domain R is a unique factorisation domain if, and only if:
1. any $r \in R \setminus \{0\}$ has a representation as $u \prod_{i=1}^{n} p_i$, where u is a unit of R and the p_i are irreducible non-unit elements of R, i.e. divisible only by their associates;
2. any such representation of r is unique up to re-ordering and multiplication by units.

It is well known that these two concepts are in fact identical.

However, we could ask for constructive versions of these.

Constructive GCD domain A constructive integral domain R is a constructive GCD domain if, and only if, there is an algorithm which, given any two elements a and b of R, computes an element g of R such that:

1. $g|a$ and $g|b$;
2. if $c|a$ and $c|b$, then $c|g$.

Constructive unique factorisation domain A constructive integral domain R is a constructive unique factorisation domain if, and only if:

1. there is an algorithm which, given any $r \in R \setminus \{0\}$ computes a representation $r = u \prod_{i=1}^{n} p_i$, where u is a unit of R and the p_i are irreducible non-unit elements of R;
2. any such representation of r is unique up to re-ordering and multiplication by units.

As we might expect, the constructive versions are stronger than the non-constructive versions. However, the two constructive versions are different: every constructive unique factorisation domain is a constructive GCD domain, but not *vice versa*. The example is given in [9]. Let λ_i be an infinite, recursively enumerable, non-recursive sequence of elements from $\{1, -1\}$, and let $R = \mathbf{Q}(\sqrt{\lambda_1}, \ldots)[x]$. Then Euclid's algorithm demonstrates that R is a constructive GCD domain. However, it is not a constructive unique factorisation domain, since the factorisation of x^2+1 tells us whether (if it factors into two linears) or not (if it is irreducible) there is a -1 in the sequence of λ_i, which was assumed to be non-recursive[2].

This problem leads to the need for clear separation of the constructive algebraic domains from the non-constructive ones, probably in different CDs, so that one could distinguish

```
<OMS name="GCDdomain" cd="domain1">
```

from

```
<OMS name="GCDdomain" cd="cdomain1">
```

(the latter being short for "constructive domains" to fit into the eight-character limit for the names of CDs).

4 What do we mean by Equality?

This problem was discussed in [7]. It was pointed out that it is necessary to be precise about the question "equal as what", since $\frac{x^2-2x+1}{x^2-1}$ and $\frac{x-1}{x+1}$ are equal as

[2] If one wishes to be more formal, one lets λ be a Kleene [15] function, i.e. a recursive function defined on all positive integers which is an injection such that $\{m : \exists n(\lambda(n) = m)\}$ is a non-recursive class. Then let $K_m = \mathbf{Q}(\sqrt{x_1}, \ldots)$, where $x_j = -1$ if j is the least n such that $\lambda(n) = m$, and $x_j = 1$ otherwise. Then $1 + x^2$ is reducible over K_m if, and only if, $\exists n(\lambda(n) = m)$, and so an effective factorisation algorithm, or even an effective irreducibility test, gives us an algorithm to test membership in a non-recursive set, a contradiction.

elements of the mathematical type $\mathbf{Q}(x)$, but not as functions $\mathbf{Q} \rightarrow \mathbf{Q}$, since the first is undefined at $x = 1$, while the latter yields 0 there. In some sense this is a typing question, in that the two expressions are equal/unequal depending on their type, but the question is in practice more fundamental than that: do we really want a flag on every expression that says "a g.c.d. has been performed, therefore you may not use `eval`"?

It is also necessary to distinguish between "exactly equal", "approximately equal" (meaning I have evaluated a certain number of digits or series terms, but have no proof of equality), "probably equal" (meaning I have performed some sort of Monte Carlo test, as in [16] for straight-line programs [8], in which case one has to ask what the probability of failure is[3]). Also does returning `false` merely mean "I am unable to prove that they are equal", as Maple does with the following

$$\frac{2}{i}\ln\left(\sqrt{\frac{1+z}{2}}+i\sqrt{\frac{1-z}{2}}\right) = -i\ln\left(z+i\sqrt{1-z^2}\right) \tag{1}$$

example.

In classical logic $(x = y) \vee (x \neq y)$ is always true, but this can cease to be true in other logics, or in algorithmic systems that do have complete decision procedures. This point is brought up in [10], where it is pointed out that, in their constructive logic, "apartness" (i.e. "I can compute, in this logic, that a and b are not equal") is the more fundamental concept.

[12–14] show that it is possible to have, at least for real algebraic numbers, a constructive implementation of $\mathbf{R}$ for which equality is decidable on this subset. It should be noted that knowing an object is, say, an integral domain in the classical sense, is often not sufficient computationally. For example, if we cannot computationally decide whether a number is zero or not, we cannot use fraction-free Gaussian elimination for finding determinants, but must use Cramer's rule or the equivalent, an $O(n!)$ algorithm rather than an $O(n^3)$ one.

The current OpenMath CDs only define one meaning of equality, which is intended to be "I can prove that a and b are equal". In some sense, this OpenMath definition is circular, since rules such as commutativity, which one might use in such a proof, are defined as $a \times b = b \times a$. Furthermore, the definition of "prove" is not stated, but, this author would argue, is implicitly that of classical logic. Other notions of equality could clearly be defined in other CDs.

These debates over the meaning of equality could be dismissed (but incorrectly in the author's view) as the hair-splitting of logicians, were it not for the fact that these distinctions crop up in practice. There has just (August 2001) been a debate on `comp.soft-sys.maple` about the difference between `FAIL` and `false` in Maple's `testeq` Monte-Carlo equality tester.

[3] [16] asks whether one should take an *a priori* or an *a posteriori* approach to this. As far as the author knows, this question has not generally been studied.

5 Just what logic are you using, anyway?

This is, in some ways, the most fundamental question of all. Many of the rules of classical logic do not hold in some constructive, or even intuitionistic, logics. At one level, this is trivially solved in OpenMath: the `logic1` CD clearly defines the operators of classical logic, and other CDs would be needed to define other concepts, e.g. a `logic3` CD to define intuitionistic logic.

However, this is the easy part. How do we know what logic was applied to yield a piece of mathematical knowledge? This is an area that is not really treated in OpenMath, where a Formal Mathematical Property is defined in the OpenMath standard [17] as follows.

> It corresponds to a theorem of a theory in *some* formal system

(our italics). Although this has not been fleshed out, it would be possible to use CDGroups in a different way to their current use (as collections of thematically-related CDs): they could be used to group together CDs containing objects and the Formal Mathematical Properties that were true in *the same* formal system.

6 So, what use is OpenMath?

In this workshop, [2] wrote

> OpenMath's Content Dictionaries are not "first class citizens". They are machine-readable, but not machine understandable; thir content is largely informal preventing the possibility of any automatic elaboration of their content. They are essentially conceived as background references for the implementors of Phrase-books.

1. Is this true?
2. Is this a bad thing?

6.1 Is this true?

OpenMath CDs in practice define various sorts of symbols.

- Abstract operations, such as "+" — not to be confused with "+ on `Nat`".
- Operations with ill-defined semantics, or multiple semantics among which an undefined choice has to be made, such as `<OMS name="int" cd="calculus1"/>` — Riemann, Riemann-Steltjes, Lebesgue ...?
  Other CDs can define more specific versions!
- Operations that could be well-defined, and probably we could try harder, such as log.
  Nevertheless, hard decisions are made, such as `logic1` havng classical semantics.
- Operations which are truly fixed by the CDs, e.g. arccot.
  These grow as we move out of the core, e.g. special functions, groups.

This is a retrospective classification by the author, and maybe these distinctions should be clearer, or even formally defined at all.

6.2 Is this a bad thing?

Clearly, the more precision OpenMath can bring, the better.
But an attempt to define the semantics of, say, "+" would restrict generality unacceptably.
Any precision is better than none, and in many key areas of ambiguity, OpenMath is faced with hard choices which it has to make.
Is guidance to phrasebook writers not worth having?

7 Conclusions

Mathematical knowledge, or at least its common representations, is less precise than is often thought — see the entries marked "inconsistent" in table 1. We have pointed out several levels of ambiguity, and noted that some of the problems are soluble through clarity of definition, as in the OpenMath Content Dictionary system. However, the fundamental ones of "what equality do you mean" and "what logic is this in" are deeper, and it is an open question whether a formal approach can encompass more than one of these properly.
However, much confusion can be clarified — "what is $\mathrm{arccot}(-1)$", "what is the Ricci tensor", "how big is D_8", "is this 'or' intuitionistic" etc.?

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. Asperti,A., Padovani,L., Sacerdot Coen,C., Guidi,F. & Schena,I., Mathematical Knowledge Management in HELM. Proc. MKM 2001.
3. Conway,J.H., Hulpke,A. & McKay,J., On Transitive Permutation Groups. *LMS J. Comput. Math.* **1** (1998) pp. 1–8. `http://www.lms.ac.uk/jcm/lms96001`.
4. Conway,J.H., Curtis,R.T., Norton,S.P. & Parker,R.A., *ATLAS of finite groups*. Oxford University Press, 1985.
5. Corless,R.M., Davenport,J.H., Jeffrey,D.J. & Watt,S.M., "According to Abramowitz and Stegun". *ACM SIGSAM Bulletin* **30** (2000) 2, pp. 58–65.
6. Corless,R.M., Davenport,J.H., Jeffrey,D.J., Litt,G. & Watt,S.M., Reasoning about the Elementary Functions of Complex Analysis. Artificial Intelligence and Symbolic Computation (ed. John A. Campbell & Eugenio Roanes-Lozano), Springer Lecture Notes in Artificial Intelligence Vol. 1930, Springer-Verlag 2001, pp. 115–126. `http://www.apmaths.uwo.ca/~djeffrey/offprints.html`. `http://link.springer.de/link/service/series/0558/bibs/1930/19300115.htm`.
7. Davenport,J.H., Equality in Computer Algebra and Beyond. Proc. Calculemus 2001 (ed. S.A. Linton) pp. 120-129. `http://www-theory.dcs.st-andrews.ac.uk/~sal/CalcCD`
8. Freeman,T., Imirzian,G. & Kaltofen,E., A System for Manipulating Polynomials Given by Straight-Line Programs. Proc. SYMSAC 86 (ACM, New York, 1986) pp. 169–175.

9. Fröhlich,A. & Shepherdson,J.C., Effective Procedures in Field Theory. *Phil. Trans. Roy. Soc. Ser. A* **248** (1955–6) pp. 407–432. Zbl. 70,35. MR 17,570d
10. Geuvers,H., Pollack,R., Wiedijk,F. &, Zwanenburg,J The algebraic hierarchy of the FTA project. Proc. Calculemus 2001 (ed. S.A. Linton) pp. 13-29. `http://www-theory.dcs.st-andrews.ac.uk/~sal/CalcCD`.
11. Gradshteyn,I.S. & Ryzhik,I.M. (ed A. Jeffrey), *Table of Integrals, Series and Products*. 5th ed., Academic Press, 1994.
12. Hur,N., *Exact Real Arithmetic in Computer Algebra*. Ph.D. Thesis, University of Bath, 2001.
13. Hur,N. & Davenport,J.H., An Exact Real Algebraic Arithmetic with Equality Determination. Proc. ISSAC 2000 (ed. C. Traverso), pp. 169–174.
14. Hur,N. & Davenport,J.H., A Generic Root Operation for Exact Real Arithmetic. Computability and Complexity in Analysis (ed. J. Blanck, V. Brattka & P. Hertling), Springer Lecture Notes in Computer Science 2064, Springer-Verlag, 2001, pp. 82–87.
15. Kleene,S.C., On Notation of Ordinal Numbers. *Journal of Symbolic Logic* **3** (1938) pp. 150–155.
16. Naylor,W.A., *Polynomial GCD Using Straight Line Program Representation*. Ph.D. Thesis, University of Bath, 2000.
17. The OpenMath Consortium. OpenMath. `http://www.nag.co.uk/projects/OpenMath/`.
18. Wolfram,S., *The Mathematica Book*. Wolfram Media/C.U.P., 1999.
19. World-Wide Web Consortium, Mathematical Markup Language (MathML [tm]) Version 2.0. W3C Recommendation of 21 February 2001. `http://www.w3.org/TR/MathML2`.
20. Zwillinger,D. (ed.), *CRC Standard Mathematical Tables and Formulae*. 30th. ed., CRC Press, Boca Raton, 1996.