Determining Empirical Characteristics of Mathematical Expression Use

Clare M. So and Stephen M. Watt

Ontario Research Centre for Computer Algebra
Department of Computer Science
University of Western Ontario
London Ontario, CANADA N6A 5B7
{clare,watt}@orcca.on.ca

Abstract. Many processes in mathematical computing try to use knowledge of the most desired forms of mathematical expressions. This occurs, for example, in symbolic computation systems, when expressions are simplified, or mathematical document recognition, when formula layout is analyzed. The decision about which forms are the most desired, however, has typically been left to the guess-work or prejudices of a small number of system designers.
This paper observes that, on a domain by domain basis, certain expressions are actually used much more frequently than others. On the hypothesis that actual usage is the best measure of desirability, this papers begins to quantify empirically the use of common expressions in the mathematical literature. We analyze all 20,000 mathematical documents from the mathematical arXiv server from 2000-2004, the period corresponding to the new mathematical subject classification. We report on the process by which these documents are analyzed, through conversion to MathML, and present first empirical results on the most common aspects of mathematical expressions by subject classification. We use the notion of a weighted dictionary to record the relative frequency of subexpressions, and explore how this information may be used for further processes, including deriving common patterns of expressions and probability measures for symbol sequences.

1 Introduction

Most software that deals with symbolic mathematical information have some pre-defined notion of when expressions are well-formed and, of the well-formed expressions, which are the most desirable. Which forms are deemed most desirable is usually decided by the software system designers, through their experience or preference, and hard-coded into the application's logic. This has made symbolic mathematical software more natural to use in some areas than others, depending on the compatibility of the system designer's choices with the user's needs. As we move toward more sophisticated, knowledge-based mathematical software, this methodology becomes increasingly problematic. In this paper we argue that it is important to understand what forms of expressions are deemed

most desirable in the actual practice of mathematics. We believe that empirical knowledge of which forms of expressions are used most often will lead to more effective mathematical software. For example, this information could be used to guide simplification in computer algebra systems, or to provide disambiguation criteria in mathematical document recognition.

Our initial motivation for this work comes from the area of mathematical handwriting recognition. We note that today's acceptable recognition rates for natural language handwriting is achieved with the aid of dictionary-based methods. For example, if the feature analysis of a stroke could yield either `Hdb` or `Hello`, then `Hello` is chosen because it is in the dictionary. At first consideration, such an approach is not suitable for mathematical handwriting recognition for several reasons: Mathematical expressions are trees, not strings. There is no fixed vocabulary from which to build a dictionary. The set of symbols alone is insufficient, and the set of possible expressions is infinite.

Nevertheless, any mathematically sophisticated person can take an arbitrary volume from a mathematical library, leaf through the pages, and, in a few seconds, have a very good idea of the precise mathematical subject area, in part, simply be noticing some characteristics of the formulae. We therefore claim that there is, in fact, usage knowledge that can and should be used by mathematical software packages. In the mathematical handwriting recognition case, this knowledge could be used to disambiguate between $\sin \omega t$ and $\sin wt$, since the former occurs much more often in practice. In the computer algebra case, this knowledge could be used to order one polynomial as x^2+1 and another as $1+\epsilon^2$.

The goals of this present line of work are to understand how

- to capture and represent empirical mathematical usage information
- to employ this information in mathematical software packages
- to analyze and organize this knowledge so as to be most useful.

We report here on our initial results toward these long-term goals. As stated earlier, we see immediate applicability to mathematical handwriting recognition and to symbolic mathematical computing. Other potential applications include mathematical searching, automated classification of mathematical documents, and mathematical data mining.

The contributions of this work are

- the identification of empirical mathematical usage as an important source of information for mathematical software design
- an approach to empirical analysis of mathematical expressions
- specific findings on symbol usage, on a subject-by-subject basis
- specific findings on most common expression usage
- methods to derive pattern expressions, and symbol-sequence Markov chains, based on analysis of instances.

#	Subject Classification
19	00 General
39	01 History and biography
228	03 Math. logic and foundations
1212	05 Combinatorics
164	06 Order, lattices, ordered alg. struct.
48	08 General algebraic systems
1383	11 Number theory
108	12 Field theory and polynomials
667	13 Commutative rings and algebras
2445	14 Algebraic geometry
240	15 Lin. and multilin. alg.; matrix thy
861	16 Associative rings and algebras
760	17 Nonassociative rings and algebras
404	18 Category theory; hom. algebra
239	19 K-theory
1169	20 Group theory and generalizations
472	22 Topological groups, Lie groups
185	26 Real functions
123	28 Measure and integration
308	30 Functions of a complex variable
59	31 Potential theory
797	32 Several complex var. & anal. spaces
312	33 Special functions
295	34 Ordinary differential equations
746	35 Partial differential equations
706	37 Dyn. systems and ergodic theory
52	39 Difference and functional eqns
21	40 Sequences, series, summability
88	41 Approximations and expansions
290	42 Fourier analysis
143	43 Abstract harmonic analysis
43	44 Integral transforms, op. calculus
34	45 Integral equations
1066	46 Functional analysis
543	47 Operator theory
164	49 Calculus of var.; optimization
171	51 Geometry
435	52 Convex and discrete geometry
1717	53 Differential geometry
226	54 General topology
627	55 Algebraic topology
1618	57 Manifolds and cell complexes
920	58 Global analysis, an. on manifolds
877	60 Prob. theory and stoch. processes
105	62 Statistics
209	65 Numerical analysis
237	68 Computer science
113	70 Mechanics of particles and systems
34	74 Mechanics of deformable solids
69	76 Fluid mechanics
13	78 Optics, electromagnetic theory
6	80 Classical thermodyn., heat xfer
553	81 Quantum theory
260	82 Stat. mechanics, struct. of matter
48	83 Relativity and gravitational theory
6	85 Astronomy and astrophysics
15	86 Geophysics
96	90 Operations research, math. prog.
42	91 Game thy, econ., soc. & behav. sci.
35	92 Biology and other natural sciences
115	93 Systems theory; control
128	94 Info. and comm., circuits
12	97 Mathematics education

Fig. 1. Count of articles by MR Subject Classification

The rest of the paper is organized as follows: We present the methodology of the current study in Section 2. As part of this study, we rely on a TEX to MathML conversion. Section 3 describes this process and extensions we have had to make for the present work. Results on frequency of symbols, as identifiers and operators, are reported in Sections 4 and 5. We present some initial results on expression analysis in Section 6. Section 7 concludes the paper.

2 Methodology

To study the empirical usage of mathematical expressions, the first step was to identify a suitable source of mathematical input. A number of possibilities existed, including

- to use logged input from a software system, such as Maple,
- to use a collection of documents from a set of cooperative authors,
- to use the articles from a particular journal

Although any of these avenues would have been easy to follow, each had its own problems: Logged input from a software system would heavily influenced by the characteristics of the system, and thus be riddled with artifacts. Articles from a small set of authors, or from a particular journal, would likely be heavily slanted in their usage and could not be taken as representative.

Instead, we chose to use the collection of articles available on the widely used, public e-Print server, `arXiv.org` [2], as our corpus of mathematical usage. This has the advantage of broad coverage by mathematical area. It also has the disadvantages that:

- Some areas are disproportionately represented.
- The mathematical material is at a research level, and this may not be representative of usage at more elementary levels.
- The material is relatively new, and is not representative of historical usage.

Bearing this in mind, we decided that the collection of articles was sufficiently representative of current mathematical usage to be useful, and that developing a collection that was more balanced by area, level, historical period, *etc*, was a long-term project.

One of the attractive properties of `arXiv.org` is its organization of articles according to the Mathematics Subject Classification, which is used to categorize items covered by the two reviewing databases, Mathematical Reviews (MR) and Zentralblatt MATH (Zbl). The current classification system, MSC 2000 [3], is a revision of the classification scheme that had been used previously by these databases. It consists of more than 5,000 two-, three-, and five-character classifications, corresponding to increasingly finely defined disciplines of mathematics. For example, "11" represents Number theory; "11B" Sequences and sets, and "11B05" Density, gaps, topology.

We followed the following steps to obtain our corpus of expressions to analyze:

The first step was to obtain all articles from `arXiv.org` from the five year period 2000–2004. This data range contained all articles since the new subject classification was introduced. To understand area-specific usage patterns, while having a sufficient number of articles in each category, we grouped articles according to their top-level, two-digit MSC classification. The count by classification of articles considered is shown in Figure 1. Altogether 22,289 articles were accessed. Of these 21,677 came with TEX source. This comprised 4.65GB of PDF files and 794 MB of TEX source.

The second step was to extract mathematical expressions from the articles. It was helpful that the articles had TEX source, but this was not usable directly for our analysis. The problems with TEX source include:

- Mathematical expressions typically use author-defined macros.
- Mathematical expressions my be hidden in macros, and not be visible in the source text.
- TEX expressions typically have only as much structure as is needed to give proper visual grouping. For example `$(ad-bc)^2$` consists of a single row of 7 items, `(`, `a`, `d`, `-`, `b`, `c` and `)^2`. Note that there is no notion that ad and bc are subexpressions, while $d - b$ is not, and note that it is only the closing parenthesis that is squared.

We used our TEX to MathML [1] converter, described in [8], to resolve these difficulties, and performed our analysis on the resulting MathML expressions.

The benefit of this approach was that the expressions treated were (for the most part) complete, well formed, and grouped appropriately. The difficulty with the approach was that not all the complexities of TEX were handled, and some expressions were incorrectly translated. However, since we are interested in the most frequently occurring expressions, the incomplete handling of infrequently occurring expressions is not, in principle, a problem. We describe the conversion process in more detail in Section 3. The overall conversion process required about three days of computer time on a personal workstation.

The third step was to examine the MathML expressions for each area, and to build three frequency tables. The first two tables contained counts of all identifier symbols (typically single letter operands) and all operator symbols. The third table counted the number of occurrences in the classification of each sub-expression. These tables were built using syntactic comparison of XML elements. For example, `<mrow><mo>(</mo><mi>a</mi><mo>)</mo></mrow>` would be treated as inequivalent to `<mfenced><mi>a</mi></mfenced>`. We therefore preprocessed the MathML to remove multiple representations for what would appear as *syntactically equivalent* mathematical expressions. This consisted of a number of simple conversions, including

- for `<mi>` and `<mo>`, normalizing the use of the `mathvariant` attribute
- for `<mfrac>`, eliminating any non-zero `linethickness` attribute
- for `<mfenced>`, convert to `<mrow>` with explicit open and close operators
- for trivial `<mmultiscripts>`, convert to `<msub>` or `<msup>`
- elimination of a number of attributes and elements related to presentation, such as spacing

3 TEX to MathML Conversion

The conversion of TEX to MathML is not a straightforward process. There is not yet a standard tool that completely solves this problem. TEX documents are, in general, programs with the computational power of a Turing machine. In practice, TEX macros are usually used to perform simple substitutions, with a smaller number performing heavy computations and transformations.

There are two principal approaches to TEX to MathML conversion: The first approach is to use alternative style files with modified definitions for the standard mathematical macros. These modified macros leave special markers in the generated `dvi` file, which are then used to generate the MathML. This approach has the advantage that all TEX files can be handled. The disadvantage is that all the high-level structure implicit in the TEX markup is discarded. This is the approach taken by `TeX4ht` [10] and the Hermes project [4].

The second approach is have a (partial) implementation of a TEX processor handle the input, and to generate MathML from the higher-level TEX operators. This has the advantage that implicit semantics in TEX markup (e.g. grouping information from braces,"`{`" and "`}`") is available to the MathML generation. The disadvantage is that, in principle, a complete TEX re-implementation is needed.

For this study, we used a TEX to MathML converter, developed within the ORCCA research group. This converter adopts the second approach. It has a partial implementation of the TEX programming language sufficient to expand the macros of interest in mathematics. Source for a TEX document may be given as a single file, or as a tree of files and using external macro packages. The correspondences between TEX and MathML are given by a set of bi-directional mapping files. These mapping files are intended to allow high-level semantic mappings between TEX and XSLT style sheets [8]. Because complex TEX macros are almost always given in style files, rather than being specified at top-level by authors, the mapping files may almost always be used to eliminate any shortcomings arising from the incomplete implementation of TEX. This translator is available on-line [5].

The conversion of all TEX source documents in the five year `arXiv.org` collection served as heavy test for the MathML converter, and a number of problems were encountered. Initially only 14,354 of the 21,677 articles could be handled automatically. First, we discovered that there were a number of TEX constructs that were not handled by the converter. The most important of these were (1) the handling of explicit positioning commands, e.g. for kerning symbols, and (2) the ability to handle arbitrary external macro packages from a search path. Dealing with these difficulties proved to be fairly easy.

The second major difficulty in the TEX to MathML translation was that a significant number of the TEX source files did not contain valid TEX. The TEX converter had been constructed assuming valid input, the idea being that an author would first produce a correct file by debugging with TEX and then, possibly long afterward, generate MathML. This assumption proved invalid — authors do not always correct their TEX errors if TEX's error recovery gives a desired output. We therefore were required to extend the TEX to MathML converter to simulate TEX error handling.

With user error handling in place, we were able to process 19,137 of the articles automatically. Of these, 19,063 were able to have their MathML canonicalized, and it is from these that we have extracted the expressions for analysis.

4 Identifiers

Our first analysis determines the most frequently occurring symbols used as identifiers in mathematical expressions. By this we mean letter-like symbols that occur as operands or function names, rather than as operators.

We counted all symbols occurring in expressions and recorded the results both for the global analysis and independently for each category. The first observation is that in each classification some symbols occur much more frequently than others, and which symbols are the most frequent differs from classification to classification.

All		
Ucode	**Id**	**Freq**
006E	n	48,150
0069	i	43,280
0078	x	36,240
006B	k	32,060
0074	t	25,967
0058	X	23,369
006A	j	23,038
0070	p	22,832
0041	A	22,791
0061	a	21,435
0064	d	19,457
006D	m	19,263
0066	f	18,235
004D	M	18,135
0073	s	17,659
0072	r	17,248
0043	C	16,915
0053	S	16,487
0047	G	16,074
03B1	α	15,943

03		
Ucode	**Id**	**Freq**
0069	i	51,565
006E	n	48,239
0078	x	41,042
0058	X	33,862
0041	A	29,845
0070	p	26,292
03B1	α	24,604
006B	k	24,374
0066	f	22,671
0061	a	22,030
0047	G	21,983
006D	m	19,893
006A	j	18,062
03C9	ω	18,015
004D	M	17,256
0053	S	17,122
0043	C	17,107
0046	F	16,773
0079	y	16,764
0074	t	15,693

11		
Ucode	**Id**	**Freq**
006E	n	58,186
0070	p	40,302
006B	k	38,230
0078	x	35,294
0069	i	35,100
0061	a	25,301
006D	m	23,642
0064	d	22,302
0071	q	21,797
0073	s	21,319
006A	j	21,153
0072	r	19,695
0074	t	19,654
0047	G	19,620
0058	X	19,535
0041	A	19,107
004B	K	18,905
0066	f	18,126
0046	F	16,524
004C	L	15,921

35		
Ucode	**Id**	**Freq**
0078	x	51,773
0074	t	49,859
0075	u	39,841
006E	n	35,705
006B	k	29,924
0069	i	28,941
0073	s	25,234
006A	j	24,968
0064	d	24,095
004C	L	21,094
03B5	ϵ	20,740
03BB	λ	20,189
0070	p	19,107
0043	C	17,450
03B1	α	17,087
0072	r	16,834
0076	v	16,820
0061	a	15,931
0079	y	15,920
0066	f	15,215

Fig. 2. The most frequent identifiers (per million) in all classifications (All), Logic (03), Number Theory (11) and Partial Differential Equations (35).

03		
Ucode	**Id**	**Freq**
03C9	ω	18,015
0046	F	16,773
0079	y	16,764
0054	T	15,605
0062	b	15,270
004B	K	15,144
0042	B	15,002
0063	c	14,586
0050	P	14,582
03BA	κ	13,285
004C	L	13,280
0056	V	12,004
0055	U	11,916
0048	H	11,452
0071	q	11,385
03B2	β	11,305
0068	h	10,369
03B3	γ	10,196
0067	g	10,104
0059	Y	9,918

11		
Ucode	**Id**	**Freq**
0071	q	21,797
004B	K	18,905
0046	F	16,524
004C	L	15,921
004E	N	15,537
0076	v	14,380
0054	T	14,126
0067	g	13,683
0050	P	13,479
007A	z	13,333
0079	y	12,880
0063	c	12,383
0048	H	12,238
0044	D	12,056
0062	b	11,867
0045	E	11,714
03C0	π	11,348
0068	h	10,550
0042	B	10,309
0075	u	10,291

35		
Ucode	**Id**	**Freq**
0075	u	39,841
004C	L	21,094
03B5	ϵ	20,740
03BB	λ	20,189
0076	v	16,820
0079	y	15,920
03BE	ξ	15,154
007A	z	14,459
0054	T	14,333
004E	N	13,906
0048	H	13,575
0052	R	12,421
0068	h	12,392
03A9	Ω	12,305
0077	w	11,562
03B4	δ	11,120
0067	g	10,933
0044	D	10,809
0071	q	10,380
03BC	μ	10,356

Fig. 3. Most frequent identifiers (per million) in Logic (03), Number Theory (11) and Partial Differential Equations (35), after excluding the 20 globally most frequent.

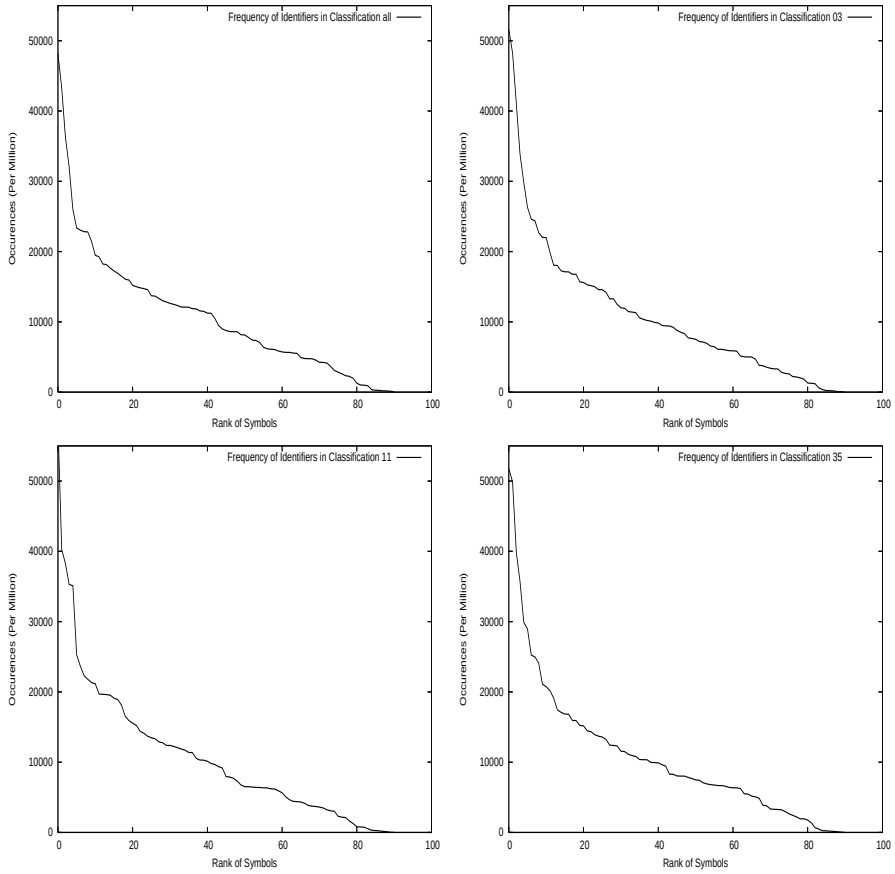

Fig. 4. Most frequent identifiers in all expressions (upper left), Logic (upper right), Number Theory (lower left), and Partial Differential Equations (lower right).
The horizontal axis gives the symbol (from most to least frequent), and the vertical axis gives the number of occurrences per million symbols in the classification.

Figure 2 shows the most frequently occurring identifiers for all the classifications taken together, as well as the most frequently occurring identifiers for three typical classifications, Logic, Number Theory and Partial Differential Equations. For detailed information on all classifications see [7].

This information could be used for disambiguation in mathematical handwriting recognition. In Number Theory, for example, we see that the letter n occurs more than twice as frequently as the letter r. By feature analysis alone, these two letters are difficult to distinguish. This frequency information is therefore useful in disambiguation.

We have arrived at a generalization of the dictionary used for disambiguation in handwriting recognition: we have constructed here, with symbols (and,

in Section 6, with expressions) a *weighted dictionary*. This structure carries information about the vocabulary of potential results, together with empirically determined weights.

Figure 3 shows the most frequently occurring identifiers for the same classifications after excluding the 20 identifiers that appear most frequently in all classifications together. We see these lists are less similar than those of Figure 2. We might use this information to aid in automatic document classification, together with word-frequency and citation analysis. Information such as this could also be used by an interactive system as a heuristic aid to determine the mathematical area in which a user is working.

Figure 4 shows, for the same classifications, the number of occurrences of identifier symbols, with the symbols ordered from most frequent to least frequent. While this will obviously be a monotonically decreasing curve, it is remarkable the degree of similarity in the shapes of these curves. We observe that although *which* symbols are used most varies quite a bit from mathematical area to area, the distribution of use of symbols is remarkably similar. In particular, after the 10% most popular identifiers, the frequency of appearance ordered by identifier decays approximately linearly.

Although, for space reasons, we have presented here the tabular results and graphs for only three classifications, and for the aggregate, the overall picture is similar for the other classifications.

5 Operators

An analogous analysis to that for identifiers was performed for operator symbols. We counted as operators anything occurring in an `<mo>` element, excluding the characters "(", ")", "[", "]", "{", "}", thinspace and underscore. We excluded the bracket forms because they were so frequent their occurrence masked the details of the other operators. Thinspace is often used for adjusting appearance, and underscores were an artifact of incomplete TEX translation. With this, Figure 5 shows the most frequently occurring operators for the same classifications as for the identifiers. Figure 6 shows the most frequently occurring operators, excluding from each category the 20 most globally common operators.

Figure 7 shows the count of operator symbols, by category, sorted from most to least popular. We note that the shape of the operator distribution is roughly similar among categories, although there are some evident differences, and even though it is different operators that are occurring most frequently. The shape of the distribution is quite different from the distribution for identifiers: generally, a few operators are used very frequently.

We see that in all areas there are a few (1-5) operator symbols that occur very frequently followed by a rapid decay in use. In particular see that more than half the symbol occurrences are from the top 10% most popular operators, and almost all occurrences are from the top 40% most popular operators.

All		
Ucode	Op	Freq
003D	=	128,715
002D	−	116,064
002C	,	112,818
2061		103,090
002B	+	79,404
2208	∋	43,942
002A	∗	29,210
2192	→	23,818
002F	/	23,405
2264	≤	20,088
02DC	˜	16,875
2297	⊗	14,242
2211	∑	13,560
003E	>	13,528
221E	∞	13,138
00AF	¯	12,451
003C	<	12,058
22EF	⋯	12,005
2202	∂	11,940
00D7	×	11,294

03		
Ucode	Op	Freq
003D	=	121,806
2061		115,262
002C	,	100,880
2208	∋	77,021
002D	−	60,732
002B	+	60,121
002A	∗	32,796
003C	<	28,345
02C9	¯	25,805
2192	→	24,370
2264	≤	24,242
002F	/	14,626
2026	…	13,495
222A	∪	12,654
2229	∩	12,483
2286	⊆	12,330
003E	>	11,784
2223	|	9,883
22EF	⋯	9,781
02DC	˜	9,428

11		
Ucode	Op	Freq
003D	=	130,735
002D	−	128,330
2061		112,484
002C	,	104,964
002B	+	94,172
002F	/	40,239
2208	∋	39,319
2211	∑	20,165
2264	≤	19,574
2192	→	18,481
002A	∗	17,757
00AF	¯	14,708
221E	∞	14,627
003E	>	12,926
22EF	⋯	12,358
02DC	˜	12,209
2265	≥	11,963
2113	ℓ	10,997
003C	<	10,151
00D7	×	10,144

35		
Ucode	Op	Freq
002D	−	138,603
002C	,	111,176
2061		103,527
003D	=	103,376
002B	+	97,579
2208	∋	38,370
2264	≤	34,575
2202	∂	28,815
002F	/	25,985
221E	∞	23,460
222B	∫	23,196
02DC	˜	19,545
003C	<	16,453
2207	∇	15,387
003E	>	15,256
002A	∗	14,470
2192	→	14,381
22C5	⋅	12,669
2211	∑	12,394
2265	≥	11,531

Fig. 5. The most frequent operators (per million) in all classifications (All), Logic (03), Number Theory (11) and Partial Differential Equations (35).
The Unicode point 2061 is the invisible "ApplyFunction" operator.

03		
Ucode	Op	Freq
02C9	¯	25,805
2026	…	13,495
222A	∪	12,654
2229	∩	12,483
2286	⊆	12,330
2223	|	9,883
2218	∘	8,894
2265	≥	8,252
2329	⟨	7,348
232A	⟩	7,072
2260	≠	6,885
2200	∀	6,390
0022	"	6,177
2227	∧	5,978
02C6	ˆ	5,825
2282	⊂	5,552
2113	ℓ	5,467
2216	\	5,282
2203	∃	4,990
22C5	⋅	4,745

11		
Ucode	Op	Freq
2265	≥	11,963
2113	ℓ	10,997
2223	|	9,474
02C9	¯	8,750
2026	…	7,829
22C5	⋅	7,728
02C6	ˆ	7,464
222B	∫	5,719
220F	∏	5,287
2282	⊂	4,938
2032	′	4,681
2260	≠	4,626
224D	≍	4,534
2229	∩	4,238
0021	!	3,692
2218	∘	3,550
2295	⊕	3,062
0022	"	2,849
00B1	±	2,796
226A	≪	2,644

35		
Ucode	Op	Freq
222B	∫	23,196
2207	∇	15,387
22C5	⋅	12,669
2265	≥	11,531
02C9	¯	9,349
02C6	ˆ	8,170
2223	|	6,379
2113	ℓ	6,074
232A	⟩	5,583
2329	⟨	5,559
00B1	±	4,556
2282	⊂	4,130
2229	∩	3,728
2272	≲	3,635
002E	.	3,375
2216	\	3,239
2260	≠	2,843
0022	"	2,767
2026	…	2,397
2032	′	2,328

Fig. 6. Most frequent operators (per million) in Logic (03), Number Theory (11) and Partial Differential Equations (35), after excluding the 20 globally most frequent.

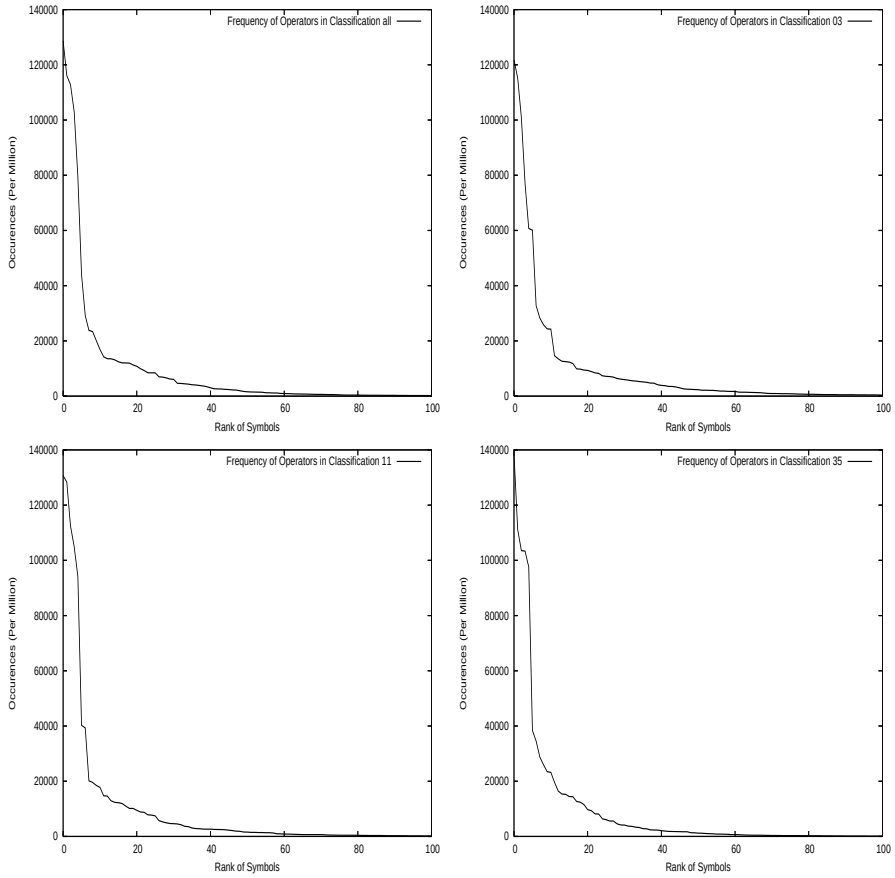

Fig. 7. Most frequent operators in all expressions (upper left), Logic (upper right), Number Theory (lower left), and Partial Differential Equations (lower right).
The horizontal axis gives the symbol (from most to least frequent), and the vertical axis gives the number of occurrences per million symbols in the classification.

We note that the shape of the distribution for the most popular operators varies by category. For example, in Number Theory and Partial Differential Equations, the first few most popular operators occur with similar frequency, followed by a sharp drop, whereas in Logic there there is a more gradual decline in frequency of use.

6 Expressions

We have performed a similar analysis for non-trivial subexpressions, counting the number of times each distinct subexpression occurs in each subject classification. The analysis of the results is more complex, however.

03				11				35			
Sz	**#**	**distinct**	**%**	**Sz**	**#**	**distinct**	**%**	**Sz**	**#**	**distinct**	**%**
2	5,151,583	13,439	.3	2	1,396,996	65,326	5	2	924,821	30,670	3
3	3,113,613	14,183	.5	3	887,089	110,311	12	3	614,469	53,193	9
4	1,703,762	14,276	.8	4	483,089	124,503	26	4	325,538	59,519	18
5	1,294,706	13,631	1.0	5	375,023	130,808	35	5	238,749	63,393	27
6	759,075	10,035	1.3	6	220,984	107,670	49	6	149,664	55,030	37
7	692,797	9,966	1.4	7	201,022	107,281	53	7	127,204	54,382	43
8	422,608	7,094	1.7	8	124,985	78,119	63	8	86,149	42,599	49
9	372,049	6,424	1.7	9	108,603	71,658	66	9	72,703	38,763	53
10	248,146	4,635	1.9	10	73,020	51,854	71	10	50,973	30,237	59
11	235,781	4,515	1.9	11	68,509	49,873	73	11	44,671	27,931	62
12	166,687	3,259	2.0	12	49,342	37,912	77	12	33,966	22,665	67
13	163,029	3,211	2.0	13	46,860	36,322	78	13	32,424	21,998	68
14	117,391	2,491	2.1	14	34,597	28,169	81	14	24,219	17,371	72
15	115,599	2,542	2.2	15	33,367	27,404	82	15	22,997	16,793	73
all	50,933,843	138,136	.27	all	14,293,554	1,362,135	9.5	all	9,613,172	802,767	8.4

Fig. 8. Number of subexpressions and of distinct subexpressions by classification and by subexpression size

#	Expression
19717	-1
15657	L^2
7903	dx
5661	t_0
4837	u_0
4752	x_0
4462	∂_t
4459	ij
4095	tx
3874	dt

#	Expression
4053	(t, x)
3399	(x, t)
2230	(x, y)
2229	$[0, T]$
1985	$-1/2$
1727	(x, ξ)
1547	$[0, 1]$
1374	(x_0)
1327	(t_0)
1206	(R^n)

#	Expression
1197	$\|x - y\|$
1163	$(n - 1)$
920	$(t - s)$
799	$(n - 2)$
733	$u(t)$
569	$(t, \cdot)$
508	$(x - y)$
499	$\frac{n-2}{2}$
496	$\|\nabla u\|^2$
441	$\Omega_0; R^3$

Fig. 9. Most frequent subexpressions of size 2 and of size 4-5 in subject classification classification Partial Differential Equations (35).

A large subexpression that occurs a certain number of times is more significant than an smaller subexpression that occurs as often, for two reasons. The first reason is that, in absolute terms, there tend to be fewer subexpressions of large size. The second reason is that there are exponentially more potential expressions of the larger size.

With the idea that the size of an expression should be part of determining the significance of its occurrences, we have analyzed each subject classification for the number of expressions. The results for subject classifications 03, 11 and 35 are shown in Figure 8. For each of these classifications and for each size, the figure shows (i) the number of subexpressions of that size that occurred in the articles, (ii) the number of *distinct* subexpressions occurring of that size, and (iii) the number of distinct subexpressions as a percentage of the number of expressions. We measure the size of an expression as the count of the following MathML tags that produce output, as opposed to providing structure. In our case, because of

the nature of the TEX to MathML conversion, the tags we counted were `<mi>`, `<mo>`, `<mn>`, `<mroot>`, `<msqrt>`, `<mfrac>`, `<menclose>`, and `<ms>`.

We observe two phenomena: First, as expected, the number of expressions occurring decreases as size increases. There are many more small expressions than large expressions. Secondly, we note that as expressions become larger, the fraction that are distinct increases. The proportion of unique subexpressions seems to depend strongly on the classification.

This analysis provides a weighted dictionary for each subject classification, providing the frequency that expressions occur in each subject classification. Space limitations preclude giving a detailed accounting of the particular expressions which occur most frequently in each classification, but we give a sample from the classification 35, Partial Differential Equations. These are shown in Figure 9. More details are available in [7].

An important question is how dependent is the weighted dictionary of subexpressions on the choice of TEX to MathML converter. Since each conversion program will have its own choices for MathML output idioms, there is a clearly dependency. However, for each expression there is a well defined collection of symbols and an intended grouping. Provided the TEX to MathML converter is consistent and provided it correctly identifies the intended groupings, the distribution of entries in the weighted dictionary should be stable under choice of converter. The application using the entries must be aware, however, that the choice of the exact way to represent a particular expression may be arbitrary.

The information in these weighted dictionaries may be used directly by applications, or may be used for further analysis. Two such directions of further analysis are deriving expression patterns, and deriving common writing sequences. We foresee many additional uses of this kind of empirical data on expression frequency.

Expression Patterns

We note that very similar subexpressions may occur frequently, for example $\sqrt{A^2+B^2}$ and $\sqrt{x^2+y^2}$. While it is possible to maintain a weighted dictionary keeping track of both of these expressions, it would be more desirable to determine that $\sqrt{\alpha^2+\beta^2}$ was a frequently occurring pattern, with suitable choices of α and β.

"Antiunification" provides an elegant framework to define such patterns. Antiunification is a process dual to unification. Rather than taking expressions and determining the most general expression to which they all can be specialized, antiunification takes a number of instance expressions and finds the least general expression which may be specialized to each instance expression. The syntactic form of antiunification has been studied since the 1970s [6].

We may determine the set of patterns from a weighted dictionary by considering all pairs of expressions. Each pair will give an antiunifier. We then consider all pairs of antiunifiers with expressions from the dictionary. These may give more antiunifiers, which are added to the set of antiunifiers. We continue to consider pairs of antiunifiers with expressions until no new antiunifiers are generated.

Since antiunification is associative, this generates a complete set of antiunifiers for the dictionary. For each antiunification, we may use the one pass algorithm of [9].

We may associate weights with these patterns simply: for each antiunifier, attempt a unification with each expression in the weighted dictionary. Then the weight of the antiunifier is the sum of the weights of the expressions with which it unifies. We note that since we are interested in syntactic expressions, this entire process of antiunification and unification is syntactic. An empirically derived, weighted dictionary of antiunifiers would provide an interesting measure to select among possibilities for "simplified" forms in a computer algebra system.

Tree-Order Symbol Sequencers

The second direction we wish to discuss for deriving expression patterns is the use of ordered tree traversals. We examine this in support of mathematical handwriting recognition. For each type of tree node, we define a traversal order corresponding to the most common writing order. For example, with

$$\sum_{i=0}^{\infty} i^2$$

the summation sign is usually written first, followed by the equation $i = 0$, then ∞, and finally i^2. Ideally the information on writing order for each node type should be determined with user experiments. Without these experiments, it is still possible to have writer-specific traversal order.

Given one or more traversal orders for each node type, we may then examine the weighted dictionary of expressions, traversing each expression, to determine Markov chains for symbol sequences. If the expression $\sum_{i=0} ...$ occurs twice as frequently as $\sum_{j=0} ...$, then the symbol sequence $\langle \Sigma, i \rangle$ gets twice the weight of $\langle \Sigma, j \rangle$. If there is not a unique traversal order for a node type, then the alternatives may be weighted.

7 Conclusions

We have proposed the idea of empirical analysis of mathematical literature as a new technique to be used in the design of sophisticated mathematical software. This is a break from the tradition of system designers using their own preferences or prejudices in determining which forms of expressions will be deemed most preferable by their systems.

We have taken presented an approach to performing empirical analysis of a body of mathematical literature. We have developed a suite of tools to convert raw TEX source to well-formed MathML, and to build weighted dictionaries of symbols and expressions.

We have made an analysis of all articles from `arXiv.org` since the new MSC 2000 subject classification. From this, we have observed that the use of mathematical symbols varies considerably from area to area and have produced usage

frequency tables for all MSC 2000 classification areas. We have observed that, while the specifics of *which* symbols are most used varies from area to area, the overall *distribution* of symbol use is very similar between areas. This is true both for symbols used as identifiers (function names and arguments), and as operators. We have also analyzed the collection of subexpressions present in the `arXiv.org` data. As well as developing a weighted dictionary for each classification area, we have observed some general properties of the frequency of distinct expressions. We are currently investigating how to best make these dictionaries available to other research projects.

Beyond these practical experiments, we have explored the potential use of information derived from symbol and expression weighted dictionaries. These have included particular applications to computer algebra, mathematical handwriting recognition and document analysis. We have also shown how weighted expression dictionaries may be used to determine further useful information, including weighted pattern dictionaries (by antiunification) and Markov chains for symbols in writing-order traversal of expression trees.

The applicability of these results depends on how representative the empirical data is. It is likely that different tables would be obtained from high-school mathematics texts, for example. Therefore, the overall approach we have taken is just as important as the specific results for this particular mathematical database.

We are excited and hopeful that the use of empirically gained knowledge may make mathematical software systems more powerful and more natural to use.

References

1. David Carlisle, Patrick Ion, Robert Miner, Nico Poppelier, Editors. Mathematical Markup Language (MathML) Version 2.0 (Second Edition). W3C Recommendation. `http://www.w3.org/TR/2003/REC-MathML2-20031021/`. October 21, 2003.
2. ArXiv e-Print Archive. `http://xxx.lanl.gov`
3. Mathematical Subject Classification (2000). American Mathematical Society. `http://www.ams.org/msc`
4. The Hermes Project. `http://alphaserv3.aei.mpg.de/hermes`
5. Ontario Research Centre for Computer Algebra. On-line TeX to MathML translator. `http://www.orcca.on.ca/MathML/texmml/textomml.html` (2002)
6. Gordon D. Plotkin. A Note on Inductive Generalization. Machine Intelligence 5 153–163 (1970).
7. Clare So. An Analysis of Mathematical Expressions Used in Practice. MSc. Thesis. University of Western Ontario. (2005)
8. Stephen M. Watt. Implicit Mathematical Semantics in Conversion between TEX and MathML, TUGBoat, Vol 23, No 1 (2002)
9. Cosmin Oancea, Clare So and Stephen M. Watt. Generalization in Maple, pp 377-382, Maple Conference 2005, Maplesoft.
10. TeX4ht: LaTeX and TeX for Hypertext, `http://www.cse.ohio-state.edu/~gurari/TeX4ht`