Symbolic Computation for Electronic Publishing

Michael P. Barnett
Department of Computer and Information Science,
Brooklyn College of the City University of New York, Brooklyn, NY 11210
Internet: `barnett@its.brooklyn.cuny.edu`

Kevin R. Perry
Interactive Computer Graphics Laboratory,
Computer and Information Technology,
Princeton University, Princeton, NJ 08540
Internet: `perry@princeton.edu`

Abstract

Recently, we developed and reported some novel ways to make MATHEMATICA produce derivations that contain conventionally structured narratives and formulas, using a procedure that interprets files containing expressions, which MATHEMATICA evaluates symbolically, control information and text. Now, this work is extended to give TEX typeset output. It supports the interactive crafting of publications that contain mechanically generated formulas and diagrams, for teaching and research, in subjects that use mathematics and other algorithmic methods. It goes beyond the direct capabilities of the built-in `Splice` and `TeXForm` functions.

1. Introduction

Recent dramatic gains in the accessibility and power of workstations has given symbolic computation greatly increased exposure. Several monographs describe MATHEMATICA (Wolfram 1991) and other computer algebra systems. Interactive use, with and without "notebooks", and batch runs that produce lengthy FORTRAN statements for numerical evaluation on mainframe computers are common. A relatively underutilized application is the combined use of electronic typesetting and symbolic computation to produce research journals, monographs, text books, reference compendia, problem books and other documents containing mathematical formulas that are:

- the result of lengthy proofs and derivations,
- burdensome to copy and check,
- numerous and closely related to each other,
- needed in different notations,
- internally repetitive (e.g. matrices whose elements are written in a common form).

Computerized symbolic calculation of the formulas in all these cases can save considerable time and effort that is needed, otherwise, to produce the results, type the manuscripts, and read and correct the proofs. The need to print mathematical formulas that were produced by primitive symbolic computations led to some of the early work on electronic typesetting over 30 years ago (Barnett 1965) — the formulas were too numerous and lengthy to derive by hand. The present paper describes some current activity that uses MATHEMATICA with plain TEX (Knuth 1986), LATEX (Lamport 1994) and AMS-TEX (Spivak 1986). It builds on our `autorun` procedure and `bilo` package (Barnett 1991; Barnett 1993; Barnett 1994), and the application of these to atomic and molecular structure calculations (Barnett 1991a; Barnett 1994a), robot kinematics (Chen 1991) and other topics.

To introduce some of the principles that underly this work, the box below depicts a workstation screen during a simple MATHEMATICA session. The default prompt for input has the form "`In[`*n*`]:=`". The value that the system computes for each input statement is displayed alongside an "`Out[`*n*`]=`" tag. Here, statement 1 assigns a simple expression to `s`. Statement 2 replaces `n` by 1 and expands the result.

```
In[1]:= s = (1 + x/(x+1))^(n+1)

                 x   1 + n
Out[1]= (1 + -----)
               1 + x

In[2]:= s /. n -> 1 // Expand

                 2
                x         2 x
Out[2]= 1 + -------- + -----
                    2   1 + x
              (1 + x)
```

MATHEMATICA evaluates:

$$a\,/.\; b \to c$$

by substituting c for b throughout a. Also, b -> c is called a "transformation rule", a "replacement rule" or just a "rule". The function f of x is written as:

$$f[x], \qquad x//f, \qquad f@x$$

These are synonymous. f is called their "head".

2. Interpreting a control file

Our typesetting work builds on the autorun procedure (Barnett 1991), that reads a file of MATHEMATICA statements and, in the default mode, mimics interactive operation. Thus, given the control file demo1 consisting of the two records:

```
s = (1 + x/(x+1))^(n+1)
s /. n -> 1 // Expand
```

a continuation of the MATHEMATICA session of Section 1 is shown in the next box. Only the autorun statement is typed. It simulates the earlier interactive action, by interpreting the control file and generating "In" and "Out" tags that combine the sequence numbers in the MATHEMATICA and autorun sessions. The screen display can be recorded, e.g., as a UNIX script file, and printed.

```
In[3]:= autorun[demo1]

In[3.1]:= s = (1 + x/(x+1))^(n+1)

                  x   1 + n
Out[3.1]= (1 + -----)
                1 + x

In[3.2]:= s /. n -> 1 // Expand

                 2
                x        2 x
Out[3.2]= 1 + -------- + -----
                     2   1 + x
               (1 + x)
```

Often, users want to see the output of the successive symbolic evaluations, without a playback of the input. autorun provides an "outputOnly" mode, and lets the control file contain formatting and related "directives" that do not get displayed, and text that does. These are labeled by # and $*$ symbols, as in the file demo2, shown in the next box. It contains two bilo formatting functions. The first reverses the reordering of the exponent — numbersLast simply moves numerical terms to the right in its target. The second treats the target as a polynomial in x.

```
# outputOnly
* Given
s = (1 + x/(x+1))^(n+1)
# format = toThe[n+1][numbersLast]
* then the expansion, when n=1, is
# format = sortToIncreasePowersOf[x]
s /. n -> 1 // Expand
```

Interpreting this file gives:

```
In[4]:= autorun[demo2]

Given

        x    n + 1
 (1 + -----)
      1 + x

then the expansion, when n=1, is

                    2
     2 x           x
1 + ----- + --------
    1 + x          2
            (1 + x)
```

3. Producing typeset output

A file demo2.tex was produced from demo2 by:

autorecord[demo2]

This invokes autorun in a mode that converts the evaluated expressions to TEX. autorecord then writes the relevant portion of the screen display to an output file, invokes LATEX or, optionally, TEX, and a previewer. Then it prints, if requested. In the TEX file that produced this paper, \input demo2 imported the TEX coded contents of the next box.

Given

$$\left(1+\frac{x}{1+x}\right)^{n+1}$$

then the expansion, when n=1, is

$$1+\frac{2x}{1+x}+\frac{x^2}{(1+x)^2}$$

The "n=1" in the second line of text is set in math mode by putting delimiters around it in the input.

A file close to demo2.tex can be produced from a slightly different control file using the built-in Splice function of MATHEMATICA, which uses the built-in TeXForm function to do the TEX encoding of individual expressions. We use the procedure toTeX, that is part of our forTeX package, to do the encoding. It provides greater flexibility and control.

4. Compound heads

Formatting the final result in demo2 involved, implicitly, the evaluation of:

`sortToIncreasePowersOf[x][1 + ...]`

In this, the head itself is a function of x.

`sortToIncreasePowersOf[x]`

We call this a "compound head". In general:

$$f[u_1, u_2, \ldots][v_1, v_2, \ldots]$$

denotes a function $g[v_1, v_2, \ldots]$, where g is the function $f[u_1, u_2, \ldots]$. Thus, the entire expression is a function of $u_1, u_2, \ldots, v_1, v_1 \ldots$ The principle is extended, e.g. in a[b][c][d][e]. Invented by Ruffini two centuries ago (see (Ruffini 1799) and (Cajori 1919, page 81)), the notation was reinvented by Curry (Curry et al. 1958; Hindley and Seldin 1986). We use it extensively, for convenience both in symbolic computation and when encoding to TEX. The ability of MATHEMATICA to handle it is very important.

5. The basic toTeX conventions

autorecord hides the TEX output of toTeX from the user. To show it here, we display a MATHEMATICA expression s, the "leads to" symbol $\rightsquigarrow$ and the typeset value of toTeX[s] as, for example, in:

`sin[theta] sin[pi+theta]` $\rightsquigarrow$ $\sin\theta \sin(\pi + \theta)$

This example, incidentally, illustrates selective parenthesization which TeXForm does not provide.

For processing by toTeX, elementary expressions are represented using standard MATHEMATICA conventions, that include == between the sides of an equation. The characters listed in (Knuth 1986, Appendix F, Tables 1-14) and many other objects are denoted by TEX-like names that omit the backslash. Thus:

alpha,	beta,	...	$\rightsquigarrow$	α,	β,	...
aleph,	hbar,	...	$\rightsquigarrow$	$\aleph$,	$\hbar$,	...
pm,	mp,	...	$\rightsquigarrow$	$\pm$,	$\mp$,	...
leq,	prec,	...	$\rightsquigarrow$	$\leq$,	$\prec$,	...

We use function notation to represent fonts, subscripts, superscripts, decorations, special bracketing and binary expressions. Thus:

`bf[A + B + C]`	$\rightsquigarrow$	$\mathbf{A+B+C}$
`P[sub[n]][sup[m]][x]`	$\rightsquigarrow$	$P_n^m(x)$
`hat[x], overline[x+y]`	$\rightsquigarrow$	$\hat{x}$, $\overline{x+y}$
`enpr[x], encr[x, y]`	$\rightsquigarrow$	(x), $\{x, y\}$
`cap[cup[A, B], C]`	$\rightsquigarrow$	$A \cup B \cap C$

enpr abbreviates "enclose in parentheses" and sapr provides parentheses that are sized automatically. Corresponding names with br, cr, lr and so forth provide square brackets, curly braces, < > and other enclosing symbols.

We use compound heads to represent several functionals. These include:

`sum[i,j,k][f[i]]`	$\rightsquigarrow$	$\sum_{i=j}^{k} f(i)$
`limit[t -> 0][h[t]]`	$\rightsquigarrow$	$\lim_{t\to 0} h(t)$
`D$[{x, 2}, y][phi]`	$\rightsquigarrow$	$\frac{\partial^3\varphi}{\partial x^2 \partial y}$

bilo has many functions that operate on these representations in symbolic calculations, e.g. rightExpand performs the reduction:

$$\sum_{i=j}^{k} f(i) \rightarrow \sum_{i=j}^{k-1} f(i) + f(k)$$

Denoting MATHEMATICA evaluation by $\Rightarrow$, the action of rightExpand is specified also by:

```
sum[i,j,k][f[i]] // rightExpand
⇒ sum[i,j,k-1][f[i]] + f[k]
```

Other bilo functions perform the reductions:

$$\sum_{i=j}^{k} f(i) \rightarrow \sum_{i=j}^{k+1} f(i) - f(k+1)$$

$$\sum_{i=j}^{k} (a+b+\ldots) \rightarrow \sum_{i=j}^{k} a + \sum_{i=j}^{k} b + \ldots$$

$$\sum_{i=j}^{k} f(i) + \sum_{i=k+1}^{l} f(i) \rightarrow \sum_{i=j}^{l} f(i)$$

and several related operations.

The compound head notation allows numerous variations, e.g. in symbolic calculations that involve integrals, we use expressions that include:

integral[x][y] $\rightsquigarrow$ $\int y\,dx$

integral[{x_1, x_2}][y] $\rightsquigarrow$ $\iint y\,dx_1 dx_2$

integral[{x_1, x_2, x_3}][y]

$\rightsquigarrow$ $\iiint y\,dx_1 dx_2 dx_3$

integral[x, u, v][y] $\rightsquigarrow$ $\int_u^v y\,dx$

integral[{{x_1, u_1, v_1}, {x_2, u_2, v_2}}][y]

$\rightsquigarrow$ $\int_{u_1}^{v_1}\int_{u_2}^{v_2} y\,dx_1 dx_2$

integral[{{x_1, u_1, v_1}, ..., {x_n, u_n, v_n}}][y]

$\rightsquigarrow$ $\int_{u_1}^{v_1} \ldots \int_{u_n}^{v_n} y\,dx_1 \ldots dx_n$

`lineIntegral[dot, L, s][bf[V]]`

$\rightsquigarrow$ $\int_L \mathbf{V} \cdot d\mathbf{s}$

`surfaceIntegral[S][f]` $\rightsquigarrow$ $\iint_S f\,d\mathbf{S}$

6. Operating on toTeX input

Other names can be converted to TEX-like names by replacement rules that precede the statements which invoke toTeX. The rules are input as directives (see Section 9). So are targeting functions that put function heads onto specific subexpressions as in:

```
    f[a] + g[b] + h[c] //
      inSuccession[
        toThe[f][bf]
        collectivelyToTerms[
          containingAny[g, h]][sabr]]
    ⇒  bf[f[a]] + sabr[g[b] + h[c]]
```

$\rightsquigarrow$ $\mathbf{f(a)} + [g(b) + h(c)]$

The targeting function here is read "apply the function bf to the part of the target that has the head f, then apply the function sabr collectively to the parts that contain either g or h." bilo contains an easily extensible set of functions that address terms, factors, elements of a list or a vector or a matrix, coefficients, arguments, parts of a fraction or a relationship, patterns and explicit items.

Suppressing parentheses: toTeX converts the brackets in an arbitrary MATHEMATICA function to parentheses. unpr elides them. Thus:

```
f[x], f[unpr[x]], f[unpr[circ]][unpr[x]]
```

$\rightsquigarrow$ $f(x),\quad f\,x,\quad f \circ x$

Varying the style: Our default style for square roots is controlled by the toTeX variable defaultSqrtStyle. Initially it is 1, with the effect:

```
sqrt[1 - sqrt[1 - Delta^2]]
```

$\rightsquigarrow$ $\sqrt{1-\sqrt{1-\Delta^2}}$

Setting it to 2 leads to output in the style:

```
sqrt[1 - sqrt[1 - Delta^2]]
```

$\rightsquigarrow$ $(1-(1-\Delta^2)^{\frac{1}{2}})^{\frac{1}{2}}$

The default can be changed by a simple MATHEMATICA assignment. Also, in the formation of the TEX codes by toTeX each sqrt[x] is converted to the intermediate form:

style[sqrt, m][x]

with m set to the value of defaultSqrtStyle that is current. An individual square root in a formula can be set in a different style n by changing it to style[sqrt, n][x] before applying toTeX. To do this, the transformation rule

sqrt -> style[sqrt, n]

is applied to the appropriate subexpression(s) by a bilo targeting function. Similar tactics are used to style other fractional powers and fractions.

Independent options: Several features in the styling of an integral can be altered independently. toTeX begins the encoding by converting the primary head integral to the form:

style[integral, a_1 -> b_1, a_2 -> b_2, ...]

The system initializes the option values b_i that are associated with the option names a_i. The statement:

setOptions[integral, a_j -> b_j, a_k -> b_k, ...]

is used to change options that, for integrals, include:

- differentialsInNumerator — defaults to no,
- allowRepeatedIntegral — no outputs multiple indefinite integrals with a single $\int$ symbol,
- variableInLower — all, multiple, and none include the variable of integration in the lower limit(s) of, respectively, all definite integrals, only multiple definite integrals, and in none,
- variableInUpper — yes and no include and omit the integration variable in the upper limit,
- includeLimits — yes and no allow and preclude the inclusion of limits.

Options are set analogously for other functionals.

7. bilo formatting functions

Rearranging an expression: Often, the default order of an expression imposed by MATHEMATICA does not give the preferred form. Non-atomic objects are represented internally by nested function expressions, for example, (a+b)(c+d) by:

```
Times[Plus[a, b], Plus[c, d]]
```

Formatting often involves rearranging the argument lists of one or more such functions, and insulating the result from automatic reordering. bilo includes a suite of sorting functions based on the action of:

$\{x_1, \dots, x_n\}$ // partByCriteria[$c_1, \dots, c_m$]

This evaluates to $\{s_1, \dots, s_{m+1}\}$ where:

1. for $v = 1$ to m, s_v is the list of x_i for which $c_1(x)$, ..., $c_{v-1}(x)$ are false and $c_v(x)$ is true,
2. s_{m+1} is the list of x_i for which $c_1(x)$, ..., $c_m(x)$ are false.

The order of the xs in each list s_v is the same as in the original list $\{x_1, \dots, x_n\}$.

The higher level sortByPresence[v_1,v_2,...] and sortByAbsence[v_1,v_2,...] are very useful. The v's can be explicit terminal subexpressions or intermediate level heads or patterns for either. Related functions include those used in Section 2.

The bilo sort functions are applied to the relevant subexpressions by targeting functions. For this kind of work, expressions containing bilo sort functions are much shorter, usually, than is possible with functions that require pairwise ordering criteria.

Further rearrangement requirements include moving the denominator of a fraction to the beginning, as in $\frac{1}{2}(x+y)$ and preventing, say, $-(a+b)$ from opening up to $-a-b$. bilo handles these.

Skeletalizing: The reduction:

$(1+x)^{10}$ // Expand // showTerms[{1, 6, -1}]

$\Rightarrow$ $1 + $«4 terms»$ + 252\,x^5 + $«4 terms»$ + x^{10}$

typifies the action of another suite of bilo functions. If p stand for a pointer list, then showTerms[p]

and showFactors[*p*] act on Plus and Times expressions, showElements[*p*] acts on lists, vectors, matrices and tensors. showArguments[*p*] acts on any function. Each show*Objects* function skeletalizes the target. By default, each shows that items are omitted by an expression of the form:

«*n object*(*s*)»

This can be overridden to display ellipses.

Insertion of codes: If *p* is a MATHEMATICA pattern that matches one of the terms in a Plus, then applying splitAfter to this Plus breaks the line after the + or − sign that follows the term. splitBefore[*p*] breaks the line before the sign.

Applying a split function to any non-atomic expression puts the subexpression which is matched onto a new line. insertBefore[*p*][*c*] is wrapped by splitBefore[*p*], where *c* is a string of TeX codes. splitAfter is handled correspondingly. Further toTeX wrappers will allow convenient control of indention and vertical spacing. Nested targeting expressions can be used to focus on the part of the target into which codes must be inserted.

8. A simple forTeX application

This relates to teaching special functions of mathematical physics. The statement:

```
autorecord[legendreAuto]
```

reads the file legendreAuto, shown in the next box, and constructs the file legendreAuto.tex.

```
* {\bf Demonstrating orthogonality.\ } We
* integrate the product of two Legendre
* polynomials of different degree. Consider
s1 = integral[x, -1, 1][P[2, x] P[4, x]]
* Substituting explicit polynomials for
* the $P_n(x)$ gives
# beginLeftAlignedGroup["="]
s2 = s1 // evaluateAndHoldSpecialFunctions
s3 = s2 // releaseAndExpandAll
# endLeftAlignedGroup
* Then term by term integration gives
# beginRunonGroup["="]
s4 = s3 // integrateTermByTermAndHold
s4 // allowEvaluation
# endRunonGroup
```

The statement \input legendreAuto in the file that set this paper produced the contents of the box in the right hand column. By default, the outputOnly mode is in effect. So is the sandwiching of individual TeX coded output expressions between centering delimiters. The first three records in the control file begin the text. The next record is the MATHEMATICA statement that assigns, to the variable s1, the integral to be evaluated. The next two records provide the next piece of text. As regards the rest of the file:

1. beginLeftAlignedGroup puts the codes to begin a left-aligned multi-formula display before the next coded expression, and an = symbol at the right. Also, it sets a switch that includes codes to continue the display, when subsequent input expressions are processed.
2. evaluateAndHoldSpecialFunctions is a short MATHEMATICA procedure, written for this application, that converts Legendre, Laguerre and other special polynomials, written as P[*n*, *x*], L[*n*, *x*], ..., to explicit polynomials that are kept separate. Here, this separation stops MATHEMATICA multiplying the denominators.
3. endLeftAlignedGroup writes the codes to end a multi-formula display and resets switches.
4. releaseAndExpandAll removes the insulation around individual parts of the target expression and applies the built-in ExpandAll function.

Demonstrating orthogonality. We integrate the product of two Legendre polynomials of different degree. Consider

$$\int_{-1}^{1} P_2(x)P_4(x)\ dx$$

Substituting explicit polynomials for the $P_n(x)$ gives

$$\int_{-1}^{1}\left(\frac{-1+3x^2}{2}\right)\left(\frac{3-30x^2+35x^4}{8}\right)dx =$$
$$\int_{-1}^{1}\left(-\frac{3}{16}+\frac{39x^2}{16}-\frac{125x^4}{16}+\frac{105x^6}{16}\right)dx$$

Then term by term integration gives

$$-\frac{3}{8}+\frac{13}{8}-\frac{25}{8}+\frac{15}{8} \quad = \quad 0$$

5. beginRunonGroup, beginLeftAlignedGroup, and endRunonGroup, endLeftAlignedGroup produce analogous effects.
6. integrateTermByTermAndHold distributes integration over the Plus in the integrand, and stops the result coalescing.
7. allowEvaluation removes the Hold around each term.

A large variety of worked examples to help teach orthogonality of special functions can be generated by applying autorecord to input files that follow the general style of this prototype, use the same bilo functions, and cycle through sets of parameters.

9. Directives

The control file directives are simply statements that use raw MATHEMATICA or invoke procedures in our `bilo`, `forTeX` and related packages. Thus, in Section2, `bilo` sorting functions are assigned to `format` — a surrogate head that is applied to each output expression before it is encoded. The directives in Section 8 are the names of TeX functions that put the delimiters for display math into the output, and control several program switches.

Statements are written as directives that:

1. substitute TeX-like names for other identifiers,
2. convert function expressions that carry indexes as ordinary arguments into representations that lead to sub- and superscripting,
3. change font and impose bracketing,
4. fine tune the style as described in Section 5,
5. perform substantive steps that are not typeset.

There is further flexibility. `autorecord` is used in the `displayBoth` mode to document the symbolic algebra part of a mechanized derivation. It plays back the input in typewriter font and the evaluated expressions as displayed mathematics. Options allow typewriter font for both. Each input expression can be run-on or left-aligned above the "$\leadsto$" symbol and conventionally displayed result. Further styles of alignment are provided for the `outputOnly` and `outputBoth` modes. Provision can be made to number the equations.

Options can be changed dynamically. Also, the directives can be made conditional on flags that are set at execution time, to vary the style of output from different runs using the same control file. The content can be varied by conditional directives that make evaluation occur silently until balancing directives are reached, or bypass text and/or evaluation completely. This can be used, e.g. to produce terse and detailed derivations from a single file, or problem sets with and without solutions.

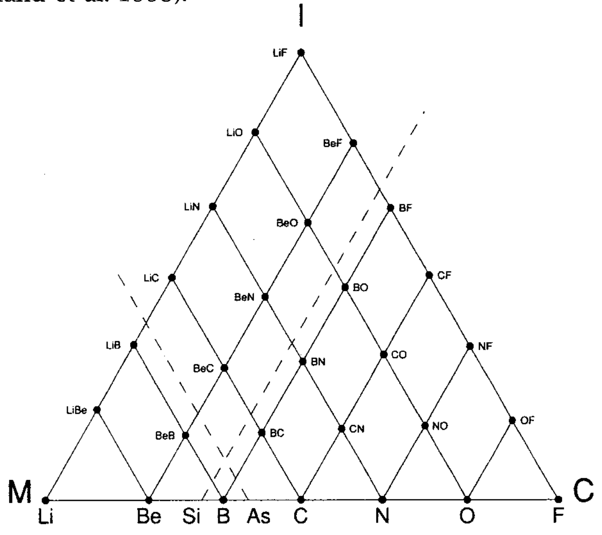
Four pre- and postprocessing functions, that default to `Identity`, which returns the argument unchanged, provide part of the flexibility.

1. `ipc` — this converts input statements into a preferred form that is recorded,
2. `ivc` — this converts input statements from a variant notation into valid MATHEMATICA,
3. `opc` — this acts on the immediate result of MATHEMATICA evaluation to give output of preferred appearance,
4. `tpc` — this operates on each block of text before it is displayed.

These parallel, in part the built-in `$PreRead`, `$Pre`, `$Post` and `$PrePrint` functions. `autorun` can document the full extent of activity by displaying any combination of the input as read, the restyled input, the input to MATHEMATICA, the output from MATHEMATICA and the restyled output, alongside tags that default to "`In`", "`in`", "`IN`", "`Out`" and "`out`", respectively.

10. Using graphics

MATHEMATICA has extensive graphics capabilities. It saves graphics objects as POSTSCRIPT files. `psfig` imports these to TeX documents and is very easy to use. The "van Arkel - Ketelaar" diagram below relates to binary compounds of the chemical elements and an atomic property called "electronegativity" (Allen, Leland et al. 1993).



I/M/C character of 1st row binaries

The letters I, M and C at the corners of the triangle stand for ionic, metallic and covalent. We wrote a MATHEMATICA procedure that reads a list of elements and produces the corresponding diagram. This example involves the algorithmic placement of text, with provision for interactive fine tuning.

The ability to import graphics helps check symbolic calculations. Computer algebra is fraught with possible error, though this is seldom discussed. The early stages of a symbolic calculation often produce results that are in the literature, and there is merit to making a comparison.

Our first efforts to check results this way showed that, often, in hand generated formulas, a particular subexpression is represented by a single variable in one place but not another. Machine generated formulas tend to be more consistent. `bilo` was developed, in part, to target the changes that are needed to bring the content and arrangement of computed results into agreement with published material.

Even when this has been done, however, the visual comparison can be tedious. For example, as part of a longer calculation (Barnett 1994a), we reconstructed a 33-term recurrence formula, that occupies half a page of Physical Review (Pekeris 1958). It can be written:

$$\sum_{(\lambda,\mu,\nu)\in S} c_{\lambda,\mu,\nu}(l,m,n)A(l+\lambda,m+\mu,n+\nu) = 0$$

where the c's are simple algebraic expressions in (l, m, n), and S consists of 33 triples whose elements are between -2 and 2. To help other workers see that the results are in agreement, the relevant passage in Physical Review was scanned, and the image dissected into about 40 pieces that contain separate terms or, in some cases, parts of terms, using a suite of image processing tools written locally. The pieces were imported into a document above the respective terms generated by MATHEMATICA, using another utility. This has been done here for terms 1-3 and 32, 33, by conversion to POSTSCRIPT and use of `psfig/tex`. Scaling is varied slightly.

$$4(l+1)(l+2)[-Z+\epsilon(1+m+n)]A(l+2,m,n)$$
$$4(l+1)(l+2)[-Z+\epsilon(1+m+n)]A(l+2,m,n)$$

$$+4(m+1)(m+2)[-Z+\epsilon(1+l+n)]A(l,m+2,n)$$
$$4(m+1)(m+2)[-Z+\epsilon(1+l+n)]A(l,m+2,n)$$

$$+4(l+1)(m+1)[1-2Z+\epsilon(2+l+m)]A(l+1,m+1,n)$$
$$4(l+1)(m+1)[1-2Z+\epsilon(2+l+m)]A(l+1,m+1,n)$$

...

$$+2ln[1-2Z+\epsilon(2m+n+1)]A(l-1,m,n-1)$$
$$2ln[1-2Z+\epsilon(2m+n+1)]A(l-1,m,n-1)$$

$$+2mn[1-2Z+\epsilon(2l+n+1)]A(l,m-1,n-1)$$
$$2mn[1-2Z+\epsilon(2l+n+1)]A(l,m-1,n-1)$$

This technique can be used to publish scanned pictures together with algorithmically constructed graphics in many areas of work.

Setting the output of symbolic computation, using special fonts and macros for chemical formulas, chess situations, musical scores and other iconic displays has still further potential. The use of built-up formulas by organic and biological chemists is well known. These, and the further needs of inorganic and solid state chemists are discussed in (Jensen 1989). We have made a start on interfacing MATHEMATICA with the work reported earlier (Haas and O'Kane 1987; Tutelaers 1992; Taupin 1992).

As a simple example of non-mathematical output produced by MATHEMATICA, two very short procedures that construct a word count and encode it as a table produced the following display from a 10-line passage in Macbeth.

a	3	adder's	1	and	8
bake	1	bat	1	blind	1
...					
trouble	2	wing	1	wool	1
worm's	1				

Distribution

The `bilo` package is in, and the `forTeX` material soon will be in the anonymous ftp library at:

`mondrian.princeton.edu (128.112.224.14)`

Acknowledgements

This work was started under support of NSF grants NAC-25001 and ASC-8500655 and conducted in part during MPB's sabbatical leave in the Chemistry Department and Interactive Computer Graphics Laboratory of Princeton University. We thank K. D. Alexander, L. C. Allen, J. F. Capitani, M. Ross, A. Shulzycki, J. S. Stamm, D. J. Thongs and P. Vince for their comments and cooperation.

References

Allen, Leland C. et al. "Van Arkel — Ketelaar triangles." *J. Molec. Struct.* **300**, pages 647-655, 1993.

Barnett, Michael P. *Computer Typesetting — Experiments and Prospects.* Cambridge, Mass.: MIT Press, 1965.

Barnett, Michael P. "Some simple ways to construct and to use formulas mechanically." *ACM SIGSAM Bulletin* **28** (2), pages 21-29, 1991.

Barnett, Michael P. "Summing $P_n(\cos\theta)/p(n)$ for certain polynomials $p(n)$." *Computers Math. Applic.* **21** (10), pages 79-86, 1991.

Barnett, Michael P. "Implicit Rule Formation in Symbolic Computation." *Computers Math. Applic.* **26** (10), pages 35-50, 1993.

Barnett, Michael P. and Kevin R. Perry. "Hierarchical Addressing in Symbolic Computation." *Computers Math. Applic.*, in press, 1994.

Barnett, Michael P. "Symbolic calculations for the He Schrödinger equation." to be published, 1994.

Cajori, Florian. *A history of mathematical notation.* vol. II, page 81, Chicago: Open Court, 1929.

Chen, Zhan-zhan. *Symbolic calculation of inverse kinematics of robot manipulators.* M.A. Thesis, Brooklyn College of the City University of New York, 1991.

Curry, Halsey B. and R. Feys. *Combinatory Logic* vol. 1, Amsterdam: Netherlands, North-Holland Publishing, 1958.

Haas, Roswithwa and Kevin C. O'Kane. "Typesetting chemical structure formulas with the text formatter TEX/LATEX." *Computers and Chemistry,* **11** (4), pages 252-271, 1987.

Hindley, J.R. and J.P. Seldin. *Introduction to Combinators and Lambda-Calculus.* New York: Cambridge University Press, 1986.

Jensen, William B. "Crystal coordination formulas." Pages 105-146 in *Cohesion and structure,* vol. 2, *The structure of binary compounds,* D. G. Pettifor and F. R. de Boer, eds. North-Holland, Amsterdam, 1989.

Knuth, Donald E. *The TEXbook*. New York: Addison-Wesley, 1986.

Lamport, Leslie *LATEX — A Document Preparation System*. 2nd edition, New York: Addison-Wesley, 1994.

Pekeris, Chaim L. "Ground state of two-electron atoms." *Phys. Rev.* **112** (5), pages 1649–1658, 1958.

Ruffini, Paolo. *Teoria generale delle equazioni*. Bologna, Italy, 1799.

Spivak, Michael D. *The Joy of TEX*. Providence, RI: American Mathematical Society, 1986.

Taupin, Daniel. "MusicTEX: using TEX to write polyphonic or instrumental music." *TUGboat* **14** (3), pages 203–211, 1993.

Tutelaers, Piet. "A font and style for typesetting chess using LATEX or TEX." *TUGboat* **13** (1), pages 85–90, 1992.

Wolfram, Stephen. *Mathematica, A System for Doing Mathematics by Computer*. 2nd edition, New York: Addison-Wesley, 1991.