HIERARCHICAL ADDRESSING USING "bilo"

MICHAEL P. BARNETT
Department of Computer and Information Science
Brooklyn College of the City University of New York
Bedford Avenue, Brooklyn, New York, 11210
and
KEVIN R. PERRY
Interactive Computer Graphics Laboratory
Computer and Information Technology
Princeton University, Princeton, NJ, 08540
June, 1993

ABSTRACT. To facilitate symbolic computation that parallels the traditional style of mathematical physics, we report some notations and supporting software, based on *Mathematica*, that let a user specify parts of a mathematical expression to be altered *in situ*. The scheme is mnemonic and easily extended. It evolved in the course of some lengthy symbolic computations that applied a variety of mathematical techniques to several areas of study. The principles are not restricted to work with *Mathematica*.

1. INTRODUCTION

1·1 Background. The informal remark "collect the products of x in the numerator" leads the reader of an elementary algebra text through the transformation

$$\frac{a\ +\ b\,x\ +\ c\,x}{d} \quad\rightarrow\quad \frac{a\ +\ (b\ +\ c)\,x}{d} \tag{1.1}$$

Specifying the parts of an expression to be altered *in situ* is an idiom as old as structured mathematical notation [1]. For input to computer sessions that perform symbolic computation, this idiom is captured in "targetting expressions", such as (1.2) for the example in hand.

$$\texttt{toTheNumerator}\big[\texttt{collectProductsOf[x]}\big] \tag{1.2}$$

Applications of *Mathematica* [2] to some studies of the electronic structure of molecules and atoms [3–5], robot kinematics [6] and a few other topics, required operations on one side or other or both sides of an equation. Operations were performed, *in situ*, on terms and factors — selectively, individually and collectively, on arguments of elementary and special functions, on elements of vectors and matrices, on variables of integration and indexes of summation, and on many other kinds of sub-expression. The kinds of operation that were used pervade the application of many mathematical techniques to science and engineering. For this work, an easily extensible collection of mnemonic "addressing functions" typified by `toTheNumerator` in (1.2), has been written, that can be applied hierarchically by nesting. They form part of the `bilo` library. An early version was reported briefly in [7] and the implied rule formation feature, which

This work began with support of NSF grants NAC–25001 and ASC–8500655. We are grateful for the cooperation of L.C. Allen and K.D.Alexander.

Typeset by $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX complements the hierarchical addressing, in [8]. As well as controlling the course of derivations precisely, in commands that are self-explanatory, the addressing functions help put the parts of output expressions into familiar arrangements for instruction, publication, and comparison with existing literature. We think this work bears on issues of symbolic computation of wider interest than just the usage of *Mathematica*.

1·2 Outline. Section 2 describes some basic notational devices and conventions that are used. Sections 3–7 describe the addressing functions that deal, respectively, with

- parts of a fraction, sides of equations and sides of other relationships,
- terms and factors of the target expression and of parts of it,
- sub-expressions which are specified explicitly, or as instances of a pattern, or by the names of functions which are their "heads" (in the sense explained in the next section),
- arguments of a function, taken individually and collectively,
- coefficients of powers, orthogonal functions and other systematically varied sub-expressions.

Just eight basic `bilo` functions, discussed in Sections 5–7, let a user define further functions with short expressive names, for wide ranging applications. Section 8 discusses the application of the methods of the earlier sections to the formatting of output. Section 9 comments on some general issues suggested by this work.

2. SOME NOTATIONAL DEVICES AND CONVENTIONS

2·1 Heads. *Mathematica* uses square brackets to denote functional dependence as in $f[x]$. We print these in different sizes, e.g. in (1.2), to emphasize nesting. *Mathematica* also supports the postfix notation $x//f$ for $f[x]$. The name of each built-in function of *Mathematica* begins in upper case. By contrast, the names of most `bilo` functions begin in lower case. We use names such as `sqrt` and `sin` for standard functions, instead of the names of the built-in *Mathematica* functions, to permit tight control of the behavior by sets of `bilo` rules. *Mathematica* uses the `=` symbol for assignment and the `==` symbol to join the two sides of an equation.

The expression (1.2) makes implicit use of the *Mathematica* equivalent of the Curry notation for functions of several variables [9]. A function f of variables x and y can be regarded as a function h of y, where h is a function g of x. Thus,

$$f(x,\, y) \;=\; h(y), \qquad h \;=\; g(x), \quad \text{and} \quad f(x,\, y) \;=\; g(x)(y) \tag{2.1}$$

Alternatively, f can be regarded as a function $\bar{h}$ of x, where $\bar{h}$ is a function $\bar{g}$ of y.

$$f(x,\, y) \;=\; \bar{h}(x), \qquad \bar{h} \;=\; \bar{g}(y), \quad \text{and} \quad f(x,\, y) \;=\; \bar{g}(y)(x) \tag{2.2}$$

In *Mathematica*, the part of a function expression that precedes the bracketed argument or arguments is called the "head". Thus `f` is the head of `f[x, y]`, `g[x]` is the head of `g[x][y]` and `g` is the head of `g[x]`. In this circumstance, we call `g[x]` a compound head.

2·2 Conversions. The effect of `collectProductsOf[x]` in the numerator of (1.1) is shown in the statement (2.3). We use the notation $a \Rightarrow b$ for "the system converts a to b automatically" where, at times, the system automatically converts b to yet another form. For present purposes, *Mathematica* and `bilo` comprise the system. For the example in hand

$$\texttt{collectProductsOf[x][a + b x + c x]} \quad \Rightarrow \quad \texttt{a + (b + c) x} \tag{2.3}$$

We denote the transformation of an expression a to an expression b by $a \;\rightarrow\; b$, as in (1.1). The value of a *Mathematica* expression of the form (2.4), which contains the open arrow `->`, is constructed from a by substituting c in place of every occurrence of b.

$$a \;/.\; b \;\texttt{->}\; c \tag{2.4}$$

The value of (2.5) is obtained correspondingly, after re-evaluating c to take note of any changes that were made to the variables on which it depends since it was assigned a value, previously.

$$a \;/.\; b \;\texttt{:>}\; c \tag{2.5}$$

The expressions b -> c and b :> c are called "transformation rules" or "replacement rules" or just "rules", and b may be a pattern (see [2], p.219). For example, given (2.6) then (2.7) follows.

$$\texttt{r} \;=\; \big(\texttt{exp[u\_] exp[v\_] :> exp[u+v]}\big) \tag{2.6}$$

$$\texttt{exp[a] exp[b] + exp[c] exp[d] /. r} \quad\Rightarrow\quad \texttt{exp[a+b] + exp[c+d]} \tag{2.7}$$

2·3 Composition and nesting. Several addressing functions are composed and nested in the next example, which uses the bilo composition function inSuccession and the *Mathematica* function Factor (see [2], p.76).

$$\left(\texttt{u}^2 + \texttt{2 u v} + \texttt{v}^2 \;==\; \frac{\texttt{b + c x + d x}}{\texttt{e + f y + g y}}\right) \quad \texttt{//}$$

$$\texttt{inSuccession}\Big[\texttt{toTheLhs[Factor]},$$

$$\texttt{toTheRhs}\Big[\texttt{toTheNumerator}\big[\texttt{collectProductsOf[x]}\big],$$

$$\texttt{toTheDenominator}\big[\texttt{collectProductsOf[y]}\big]\Big]\Big]$$

$$\Rightarrow \quad \left(\texttt{(u + v)}^2 \;==\; \frac{\texttt{b + (c + d) x}}{\texttt{e + (f + g) y}}\right) \tag{2.8}$$

When f_1, f_2, ... , f_n stand for function heads,

$$\texttt{inSuccession[}f_1, f_2, \ldots, f_n\texttt{][}s\texttt{]} \quad\Rightarrow\quad s \texttt{ // } f_1 \texttt{ // } f_2 \texttt{ // } \ldots \texttt{ // } f_n \tag{2.9}$$

This has the same value as the built-in Composition[f_n, ... , f_2, f_1][s], (see [2], p.213). The inSuccession function, however, also allows rules and lists of rules as arguments. Each is treated as the replacement function which applies it. Thus, in the next example, the Expand action gives exp[x] exp[y] + exp[x] exp[z]. Then the rule r condenses each product of two exp factors.

$$\texttt{exp[x] (exp[y] + exp[z]) // inSuccession[Expand, r]} \;\Rightarrow\; \texttt{exp[x+y] + exp[x+z]} \tag{2.10}$$

Also, if z stands for a function head or a rule or a list of rules, the inSuccession function treats z^n as n successive arguments z, when n is a positive integer, and applies z repeatedly, until it makes no further change, when n is the bilo keyword totally. The name "inSuccession" was chosen, rather than an explicit modification of the word "composition", for conciseness and to maintain the appearance of natural language phraseology.

The functions toTheNumerator and toTheDenominator, and all the further addressing functions to be discussed, apply the inSuccession function of their argument lists to the appropriate parts of the target expression. The accommodation of rules by the inSuccession and the to ... functions is very powerful when combined with the methods of [8].

3. TARGETTING PARTS OF FRACTIONS AND RELATIONSHIPS

3·1 Fractions. The three bilo functions that address parts of a fraction are

$$\texttt{toTheNumerator}, \qquad \texttt{toTheDenominator}, \qquad \texttt{toTheNumeratorAndDenominator} \tag{3.1}$$

The first two appear in (1.2) and (2.8). The third is illustrated by the conversion (3.2). This was needed in the derivation of equation (3.8) of [4]. It is performed by (3.3), which contains the bilo function with the self-explanatory name factorOut.

$$\frac{\left(c^{k-n} \,-\, b^{k-n}\right) q_n}{k \,-\, n} \quad\longrightarrow\quad \frac{\left(b^{k-n} \,-\, c^{k-n}\right) q_n}{n \,-\, k} \tag{3.2}$$

$$\texttt{toTheNumeratorAndDenominator}\big[\texttt{factorOut[-1]}\big] \tag{3.3}$$

3·2 Equations. The three bilo functions that address sides of an equation are

$$\texttt{toTheLhs}, \qquad \texttt{toTheRhs}, \qquad \texttt{toBothSides} \tag{3.4}$$

The first two appear in (2.8). The third is illustrated by the conversion of (3.5) to (3.6). The $i_n(z)$ are related to the Bessel functions $I_{n+\frac{1}{2}}(z)$ and are used in [3].

$$\sqrt{\frac{2}{\pi}}\frac{z^{n+\frac{1}{2}}}{(2n+1)!!}\,i_n(z) \quad = \quad \sqrt{\frac{2}{\pi}}\frac{z^{n-\frac{3}{2}}}{(2n-3)!!}\,i_{n-2}(z) \;-\; \sqrt{\frac{2}{\pi}}\,(2n-1)\,\frac{z^{n-\frac{3}{2}}}{(2n-1)!!}\,i_{n-1}(z) \tag{3.5}$$

$$z^2\,i_n(z) \quad = \quad \frac{(2\,n+1)\,!!}{(2\,n-3)\,!!}\,i_{n-2}(z) \;-\; (2\,n-1)\,\frac{(2\,n+1)\,!!}{(2\,n-1)\,!!}\,i_{n-1}(z) \tag{3.6}$$

The targetting expression (3.7) that performs the conversion contains `times[`u`][`v`]`. This is a `bilo` function that returns $v \times u$. The `bilo` function `sqrtSimplify` simplifies products of `sqrt` functions. The double factorial $n!!$ is represented by `bb[n]`.

$$\texttt{inSuccession}\Big[\texttt{toBothSides}\Big[\texttt{times}\big[\texttt{z}^{\texttt{3/2-n}}\,\texttt{sqrt}\left[\frac{\texttt{pi}}{\texttt{2}}\right]\,\texttt{bb[2n+1]}\big]\Big], \\ \texttt{toTheRhs[Distribute, sqrtSimplify]}\Big] \tag{3.7}$$

Applying `Distribute` to the left side is redundant but harmless, so (3.8) has the same effect.

$$\texttt{toBothSides}\Big[\texttt{times}\big[\texttt{z}^{\texttt{3/2-n}}\,\texttt{sqrt}\left[\frac{\texttt{pi}}{\texttt{2}}\right]\,\texttt{bb[2n+1]}\big],\ \texttt{Distribute, sqrtSimplify}\Big] \tag{3.8}$$

3·3 Another example. The conversion of (3.9) to (3.10) applies a function that persists in the final result.

$$e^{a\,+\,b} \quad = \quad e^a \;\times e^b \tag{3.9}$$

$$\log(u\,v) \quad = \quad \log u \;+\; \log v \tag{3.10}$$

In the style of [8], we begin the conversion by making the assignments

$$\begin{aligned} &\texttt{eqn[1]} \;\; = \;\; \texttt{u} \;\; \texttt{==} \;\; \texttt{exp[a]} \\ &\texttt{eqn[2]} \;\; = \;\; \texttt{v} \;\; \texttt{==} \;\; \texttt{exp[b]} \\ &\texttt{eqn[3]} \;\; = \;\; \texttt{exp[a + b]} \;\; \texttt{==} \;\; \texttt{exp[a]} * \texttt{exp[b]} \\ &\texttt{eqn[4]} \;\; = \;\; \texttt{log} \;\; \texttt{==} \;\; \texttt{InverseFunction[exp]} \end{aligned} \tag{3.11}$$

`InverseFunction` is built-in (see [2], p.200). `bilo` assigns `brule[4]` the value (3.12), (see [8]).

$$\texttt{log :> exp}^{\texttt{-1}} \tag{3.12}$$

For conciseness, a name is assigned to the targetting expression that is used next.

$$\texttt{localFunction} \;\; = \;\; \texttt{inSuccession}\Big[\texttt{toBothSides[log], toTheRhs}\big[\texttt{brule[4]}\big]\Big] \tag{3.13}$$

`bilo` reduces `inSuccession[`$\ldots,\, a,\, f^{-1},\, f,\, b,\, \ldots$`]` to `inSuccession[`$\ldots,\, a,\, b,\, \ldots$`]`. Hence,

$$\begin{aligned} &\texttt{eqn[5]} \;\; = \;\; \big(\texttt{eqn[1] // localFunction}\big) \;\; \Rightarrow \;\; \big(\texttt{log[u]} \;\; \texttt{==} \;\; \texttt{a}\big) \\ &\texttt{eqn[6]} \;\; = \;\; \big(\texttt{eqn[2] // localFunction}\big) \;\; \Rightarrow \;\; \big(\texttt{log[v]} \;\; \texttt{==} \;\; \texttt{b}\big) \end{aligned} \tag{3.14}$$

The `bilo` function `reverse` switches the two sides of an equation, and `bilo` assigns

$$\texttt{exp[a] :> u,} \qquad \texttt{exp[b] :> v,} \qquad \texttt{a :> log[u],} \qquad \texttt{b :> log[v]} \tag{3.15}$$

to `bruleReverse[1]`, `bruleReverse[2]`, `bruleReverse[5]` and `bruleReverse[6]`. Hence,

$$\begin{aligned} &\texttt{eqn[3] // inSuccession}\Big[\texttt{toTheRhs}\big[\texttt{bruleReverse[1], bruleReverse[2]}\big], \\ &\qquad \texttt{toBothSides[log],} \\ &\qquad \texttt{toTheLhs}\big[\texttt{brule[4], bruleReverse[5], bruleReverse[6]}\big], \\ &\qquad \texttt{reverse}\Big] \\ &\Rightarrow \;\; \texttt{log[u v]} \;\; \texttt{==} \;\; \texttt{log[u] + log[v]} \end{aligned} \tag{3.16}$$

A similar process effects the reverse derivation, and both can be put in a common form.

3·4 Related usage. The functions `toTheLhs`, `toTheRhs` and `toBothSides` also act on expressions containing the *Mathematica* operators shown in (3.17) and on further binary operators defined in `bilo`.

$$\texttt{<} \qquad \texttt{<=} \qquad \texttt{>=} \qquad \texttt{>} \qquad \texttt{->} \qquad \texttt{:>} \tag{3.17}$$

4. TARGETTING TERMS AND FACTORS

4·1 Basic constructions. We call an expression of the form $a + b + \ldots$ a "Plus" and an expression of the form $a \times b \times \ldots$ a "Times", as in *Mathematica*, thereby reserving "sum" and "product" for expressions that begin with Σ and Π. The principal `bilo` functions that operate on the terms of a Plus are

1. `toEachTerm[`*action*`][`*target*`]`, that acts on each of the terms, individually,
2. `toEachTermThat[`*selector*`][`*action*`][`*target*`]`, that acts on the selected terms, individually,
3. `collectivelyToTheTermsThat[`*selector*`][`*action*`][`*target*`]`, that acts on the Plus of the selected terms.

The several forms that *selector* can take are described at the end of the section.

4·2 Non-selective application. The usage of `toEachTerm` is shown in the derivation of a recurrence formula for $x^2\, f_n(x)$ from

$$x\, f_n(x) \quad = \quad n\, f_n(x) \;+\; (n\,+\,1)\; f_{n-1}(x) \tag{4.1}$$

This is the prototype of a derivation involving Laguerre functions $L_n(x)$ needed in [5]. Multiplying throughout by x, and using (4.1) to replace $x\, f_n(x)$ and $x\, f_{n-1}(x)$ gives

$$x^2\, f_n(x) \;=\; n \left\{ n\, f_n(x) \,+\, (n+1)\, f_{n-1}(x) \right\} \,+\, (n+1) \left\{ (n-1)\, f_{n-1}(x) \,+\, n\, f_{n-2}(x) \right\} \tag{4.2}$$

For the mechanization of this step see the discussion of (4.24) in [8]. Distributing the multipliers n and $(n+1)$ onto the rest of the terms which they begin gives (4.3). This is done by (4.4). The `bilo` function `leftDistribution` distributes the leftmost factor of a Times onto the terms of the nearest factor that is a Plus..

$$x^2\; f_n(x) \;=\; n^2\; f_n(x) + n\;(n+1)\; f_{n-1}(x) + (n-1)\;(n+1)\; f_{n-1}(x) + n\;(n+1)\; f_{n-2}(x) \tag{4.3}$$

$$\texttt{toTheRhs}\big[\texttt{toEachTerm[leftDistribution]}\big] \tag{4.4}$$

4·3 Selective application. To illustrate the usage of the function that acts on terms of a Plus that are selected by dint of their content, we refer to the derivation of equation (3.17c) of [4]. This leads, at one point, to

$$E_{0,3}(\theta) \;=\; -\frac{3}{2} \;+\; 2\,\left(\frac{3}{2}\,\cos\theta \;+\; \frac{1}{2}\right)\,\sin\frac{\theta}{2} \;+\; P_2\,(\cos\theta)\,\log\left(1 \;+\; \csc\frac{\theta}{2}\right) \tag{4.5}$$

The corresponding expressions for the $E_{0,k}(\theta)$, $k > 3$, contain $P_{k-1}(\cos\theta)$ times the log item and longer multinomials in $\cos\theta$ and $\sin\frac{\theta}{2}$. To replace $\cos\theta$ by $1 - 2\sin^2\left(\frac{\theta}{2}\right)$ except in the Legendre function, leading to a polynomial in $\sin\frac{\theta}{2}$ plus the log term, we use

$$\texttt{toEachTermThat}\big[\texttt{doesNotContain[P]}\big]\left[\texttt{cos[theta] -> 1 - 2 sin}\left[\frac{\texttt{theta}}{2}\right]^2\right] \tag{4.6}$$

Also, the derivation of (3.18c) of [4] leads to

$$\begin{aligned} E_{3,-2}(\theta) \;=\; & \frac{1}{4}\left(1 \,-\, 7\,\cos^2\theta\right) \;-\; 3\,\cos\theta\,\sin\frac{\theta}{2} \;-\; \sin\frac{\theta}{2} \;+\; \frac{1}{2}\,P_0(\cos\theta) \\ & +\, P_1(\cos\theta) \;-\; P_2(\cos\theta)\,\log\left[\sin\frac{\theta}{2}\,\left(1 \,+\, \sin\frac{\theta}{2}\right)\right] \end{aligned} \tag{4.7}$$

The corresponding expressions for all $E_{k,1-k}(\theta)$, $k \geq 3$, contain terms in $P_0(\cos\theta), \ldots\, P_{k-2}(\cos\theta)$. The `bilo` function `evaln[P]` converts $P_m(\cos\theta)$, for integer m, into an explicit polynomial in $\cos\theta$. This `bilo` function is applied to all the Legendre functions, except the multiplier of the log item, by operating with (4.8) on the entire expression.

$$\texttt{toEachTermThat}\big[\texttt{doesNotContain[log]}\big]\big[\texttt{evaln[P]}\big] \tag{4.8}$$

4·4 Using pattern matching. To illustrate the usage of the function that acts on terms of a Plus that are selected by pattern matching, we refer to some work on "molecular integrals". To validate the procedures that were developed for an extension of [3], we reconstructed the formulas for some integrals over the space V of a point distant r_a and r_b from origins A and B, that were published in a table that has had widespread use [10]. In the notation of this table, α and β are constants, ρ is the distance AB and

$$J_{13} \quad = \quad \int r_a\, r_b\, e^{-\alpha\, r_a\, -\, \beta\, r_b}\, dV \tag{4.9}$$

The mechanical derivation produced the integrals in a uniform style that, for J_{13}, contains the expression (4.10), in one formulation, where X is given by (4.11).

$$s \quad = \quad 192\,\beta^4 \;-\; 48\,\beta^2\,(\beta\,\rho \,-\, 4)\,X \;+\; 4\,(\beta^2\,\rho^2 \,-\, 10\,\beta\,\rho \,+\, 5)\,X^2 \;+\; 3\,\rho^2\,X^3 \tag{4.10}$$

$$X \quad = \quad \alpha^2 \,-\, \beta^2 \tag{4.11}$$

The formulas in [10] vary considerably in the extent to which α, β and X are eliminated. We begin the conversion of (4.10) to the published form by removing the parentheses, replacing X in the terms $192\,\beta^2\,X$ and $20\,X^2$ that result, and replacing X^n by $(\alpha^2 - \beta^2)\,X^{n-1}$ in the terms $-40\,\beta\,\rho\,X^2$ and $3\,\rho^2\,X^3$. The resulting expression is collected on powers of X, and the terms in X and X^2 are factored. The totality of these actions gives

$$20\,\alpha^4 \;+\; 152\,\alpha^2\beta^2 \;+\; 20\,\beta^4 \;-\; 8\,\beta\,(5\,\alpha^2 \,+\, \beta^2)\,\rho\,X \;+\; (3\,\alpha^2 \,+\, \beta^2)\,\rho^2\,X^2 \tag{4.12}$$

The targetting expression (4.13) converts (4.10) to (4.12). It uses `matches[`*pattern*`][`*expr*`]`, a `bilo` function which simply wraps the built-in `MatchQ[`*expr*, *pattern*`]` (see [2], p.228). The patterns `192 * (_)`, `20 * (_)`, `40 * (_)` and `3 * (_)`, provide an *ad hoc* identification of the relevant terms. The `|` notation is used for `Alternatives` (see [2], p.229). The parentheses, `*` and order of factors in the patterns are provided for readability but are not required by the system.

```
inSuccession[Expand,
        toEachTermThat[matches[192*(_) | 20*(_)]][X -> alpha^2 - beta^2],
        toEachTermThat[matches[40*(_) | 3*(_)]][X^n_ -> X^(n-1) (alpha^2 - beta^2)],
        collectPowersOf[X],
        toEachTermThat[contains[X]][Factor],
        toEachTermThat[doesNotContain[X]][Expand]]                          (4.13)
```

4·5 Collective action. We illustrate the usage of `collectivelyToTheTermsThat` by the conversion of (4.12) to the form (4.14) that appears in [10], by use of (4.15).

$$4(\,5\,\alpha^4 \;+\; 38\,\alpha^2\,\beta^2 \;+\; 5\,\beta^4) \;-\; 8\,\beta\,(5\,\alpha^2 \,+\, \beta^2)\,\rho\,X \;+\; (3\,\alpha^2 \,+\, \beta^2)\,\rho^2\,X^2 \tag{4.14}$$

```
collectivelyToTheTermsThat[doNotContain[X]][Factor]                         (4.15)
```

Another example is the conversion of (4.3) to

$$x^2\,f_n(x) \;=\; n^2\,f_n(x) \;+\; (n\,+\,1)\,(2\,n\,-\,1)\,f_{n-1}(x) \;+\; n\,(n\,+\,1)\,f_{n-2}(x) \tag{4.16}$$

This is done by factoring the collection of terms that contain $(n-1)$, using

```
collectivelyToTheTermsThat[contain[n-1]][Factor]]                           (4.17)
```

The `bilo` functions that address factors within a "Times" parallel those that address terms within a Plus. We illustrate the usage of `toEachFactor` and `toEachFactorThat` in Sections 7 and 8. The usage of the function that acts on selected factors collectively is shown by the conversion (4.18). It is effected by (4.19). The $i_n(z)$ functions were introduced for (3.5) and (3.6).

$$\frac{(2\,n\,-\,1)\,(2\,n\,+\,1)}{z^2}\,\big(i_{n-2}\,(z)\,-\,i_{n-1}\,(z)\big) \quad\longrightarrow\quad \frac{4\,n^2\,-\,1}{z^2}\,\big(i_{n-2}\,(z)\,-\,i_{n-1}\,(z)\big) \tag{4.18}$$

```
collectivelyToTheFactorsThat[doNotContain[z]][Expand]                       (4.19)
```

4·6 Unique selection. It happens quite often in a derivation that only a single term or factor can be picked by the *selector*. Although the functions toEachTermThat and toEachFactorThat work perfectly well in this circumstance, the aliases toTheTermThat and toTheFactorThat provide more readable discourse. Just as the toEach ... functions will pick just a single item, when only one meets the selection requirements, so the toThe ... functions pick several, when these meet the requirements. This is in accordance with the object of the aliases being to facilitate stylistic consistency rather than to enforce it.

To illustrate the usage of toTheTermThat, we refer to the conversion (4.20), where α and β stand for long expressions free of π. This occurred when deriving equation (5.16) of [4].

$$\frac{2^{2j}}{4}\,\pi\,\alpha \;+\; \frac{2^{2j}}{2}\,\beta \qquad \longrightarrow \qquad 2^{2j-2}\,\pi\,\alpha \;+\; 2^{2j-1}\,\beta \tag{4.20}$$

The rule (4.21) is used within the targetting expression that performs this conversion.

$$\texttt{powerDown} \;\;=\;\; \left(\texttt{2}^{\texttt{n\_}} \texttt{ :> 2 * } \texttt{2}^{\texttt{n-1}}\right) \tag{4.21}$$

This rule is applied twice to the term containing π and once to the other term by (4.22).

$$\begin{aligned}\texttt{inSuccession}\Big[&\texttt{toTheTermThat}\big[\texttt{contains[pi]}\big]\,\texttt{[powerDown}^{\texttt{2}}\texttt{]},\\ &\texttt{toTheTermThat}\big[\texttt{doesNotContain[pi]}\big]\,\texttt{[powerDown]}\Big]\end{aligned} \tag{4.22}$$

To illustrate the usage of the function toTheFactorThat, we refer to the conversion (4.23). This models several steps in the derivation of equation (3.8) of [4]. It is effected by (4.24).

$$(1 \;-\; \cos\theta)\; P_n(\cos\theta) \qquad \longrightarrow \qquad 2\,\sin^2\frac{\theta}{2}\; P_n(\cos\theta) \tag{4.23}$$

$$\texttt{toTheFactorThat}\big[\texttt{doesNotContain[P]}\big]\left[\texttt{cos[x\_] :> 1 - 2}\left(\texttt{sin}\Big[\frac{\texttt{x}}{\texttt{2}}\Big]\right)^{\texttt{2}}\right] \tag{4.24}$$

4·7 Selectors and aliases. In general, *selector* may be

1. a unary function which, when applied to a sub-expression, returns either true or false,
2. a pointer q or a list of pointers $\{q_1,\, q_2,\, \ldots\}$, where minus signifies counting from the right,
3. a logical expression in which the atoms are *selector*s of type 1,
4. a function, such as innermost and outermost, that identifies one of a set of sub-expressions,
5. a sequence $s_1,\, s_2,\, \ldots$ of items of kinds 1 to 4, that narrow the selection successively,
6. a space or a null string or Null, which signify no restriction.

The unary functions that have been used most often as selection criteria are

$$\begin{array}{lll}\texttt{contains[}x\texttt{]}, & \texttt{containsAny[}x_1,\, x_2,\, \ldots\texttt{]}, & \texttt{containsAll[}x_1,\, x_2,\, \ldots\texttt{]},\\ \texttt{matches[}v\texttt{]}, & \texttt{isHeadedBy[}h\texttt{]} & \end{array} \tag{4.25}$$

and the corresponding negations, that begin doesNot and isNot. Here, x, x_1, ... may be explicit terminal sub-expressions, or heads, or patterns to be matched by either; v can be any pattern to be matched by one or more terminal sub-expressions, and h can be any head or a pattern to be matched by one or more heads.

The full sets of aliases for the three kinds of function that address terms are

```
toEachTerm,       toTheTerm                                                (4.26)
toEachTermThat,   toTheTermsThat,   toTheTermThat,   toTerms,   toTerm     (4.27)
collectivelyToTheTermsThat,         collectivelyToTerms                    (4.28)
```

We provide all these because, for example, in a derivation which

1. in one stage, deals with each term without exception, and
2. in a later stage, deals with each term that meets a non-unique selection,

it is natural to use toEachTerm[*action*] first, and then to use toEachTermThat[*selector*][*action*].

In contrast, in a derivation which

1. in an early stage, deals with a term satisfying a unique criterion,
2. in a later stage, deals with terms satisfying a criterion that is more general, and
3. still later, deals with all the terms without qualification,

a natural sequence is `toTheTermThat[`*selector*`][`*action*`]`, `toTheTermsThat[`*selector*`][`*action*`]` and then `toEachTerm[`*action*`]`. Also, it is natural to omit the article when using a pointer or pointer list, as in `toTerm[1][`*action*`]` and `toTerms[{1,3}][`*action*`]`. Aliases for (4.25) that begin, for example, with `contain`, `containing`, `match` and `matching`, and corresponding negations, are provided for grammatical consistency. Aliases analogous to (4.26)–(4.28) are provided for all the functions that deal with factors.

Pointers are allowed, as in `toTerm[1]`, primarily for consistency with some generalizations in Sections 5 and 6. The use of pointers to terms and to factors is hampered by its dependence on the ordering within the internal representation imposed by *Mathematica*. This is not obvious for the factors of a fraction. Also, the order often is changed by operations on the terms and factors. The `bilo` function `numberTheTerms` converts an expression of the form $x_1 + x_2 + \ldots$ to

$$\texttt{term[1]}[x_1] + \texttt{term[2]}[x_2] + \ldots \tag{4.29}$$

This can be handled by functions described in the next section, and the heads `term[`n`]` removed by the `bilo` function `unnumberTheTerms`. The functions `numberTheFactors` and `unnumberThe-Factors` act correspondingly on factors of a `Times`.

5. THE BASIC BILO INFRA-STRUCTURE

5·1 The primitives. We introduce the basic `bilo` addressing mechanism by a function that has the aliases

$$\texttt{toEach}[\mathit{typifier}][\mathit{action}][\mathit{target}], \qquad \texttt{toThe}[\mathit{typifier}][\mathit{action}][\mathit{target}] \tag{5.1}$$

These generalize (4.26). Here, and elsewhere in this paper, *typifier* may be

1. an explicit sub-expression, as in (5.2), (5.4) and (5.7) below,
2. the head of a sub-expression, as in (5.13) and (5.17) below.
3. a pattern matched by a head or by a terminal sub-expression, as in (5.19), (5.21) and (5.22).

The expressions (5.1) are evaluated by operating, *in situ*, with `inSuccession[`*action*`]` on each sub-expression of *target*, which is matched or that has a head which is matched. The system does not check for consistency between use of `The` or `Each` and the number of matched items. As with the functions that address terms and factors, the form `toThe` is provided for naturalness when it will be apparent to the reader that only one instance of the specified sub-expression occurs.

5·2 Explicit typifiers. The function (5.2) shows the usage of an explicit sub-expression as *typifier*. It uses the `toThe` alias, performs the conversion (4.23) and is a little shorter than the function (4.24) given earlier.

$$\texttt{toThe}\big[\texttt{1 - cos[theta]}\big]\Big[\texttt{cos[x\_] :> 1 - 2}\Big(\texttt{sin}\Big[\frac{\texttt{x}}{\texttt{2}}\Big]\Big)^{2}\Big] \tag{5.2}$$

For another simple example, we refer to (5.3). This occurred in [5].

$$x^2\, L_n''(x) = -n\,(n+1)\,L_{n+1}(x) \;+\; n\,(3n\,+\,1)\,L_n(x) \\ +\; n\,(1\,-\,3n)\,L_{n-1}(x) \;+\; n\,(n-1)\,L_{n-2}(x) \tag{5.3}$$

The coefficient of $L_{n-1}(x)$ is converted to $-n\,(3n-1)$ by applying (5.4) to the entire equation.

$$\texttt{toThe[(1-3n)]}\big[\texttt{factorOut[-1]}\big] \tag{5.4}$$

A slightly longer example converts (3.6) to (5.5). It uses the `toEach` and the `toThe` aliases.

$$z^2\, i_n\,(z) \quad = \quad (2\,n-1)\,(2\,n+1)\,i_{n-2}\,(z) \;-\; (2\,n-1)\,(2\,n+1)\,i_{n-1}\,(z) \tag{5.5}$$

In the targetting expression we use the rule (5.6) that corresponds to $n!! = n \times (n-2)!!$.

$$\texttt{r} \quad = \quad \texttt{bb[n\_] :> n * bb[n-2]} \tag{5.6}$$

The conversion is performed by applying (5.7) to (3.6).

$$\texttt{toTheRhs}\Big[\texttt{toEach}\big[\texttt{bb[2n + 1]}\big]\texttt{[r]}\texttt{, toThe}\big[\texttt{bb[2n - 1]}\big]\texttt{[r]}\Big] \tag{5.7}$$

The first item in (5.7) converts the right hand side of (3.6) to

$$\frac{(2\,n+1)\,(2\,n-1)\,!!}{(2\,n-3)\,!!}\;i_{n-2}\,(z) \;-\; (2\,n-1)\,(2\,n+1)\;i_{n-1}\,(z) \tag{5.8}$$

The second item in (5.7) then acts on the $(2n-1)!!$ in this intermediate result, to give (5.5). The head `toTheRhs` is used only to compose the two items in its brackets — `inSuccession` works just as well. Another way to reduce (3.6) to (5.5) is described in Section 5 of [8].

5·3 Heads as typifiers. To illustrate the usage of a head as *typifier*, we refer to the Schrödinger equation for the helium atom, expressed in terms of the distances r_1 and r_2 of the two electrons from the nucleus and the inter-electron distance r_{12}. In [5], it is transformed to the perimetric coordinates

$$u \;=\; \epsilon\,(r_2 \;-\; r_1 \;+\; r_{12}), \qquad v \;=\; \epsilon\,(r_1 \;-\; r_2 \;+\; r_{12}), \qquad w \;=\; 2\,\epsilon\,(r_1 \;+\; r_2 \;-\; r_{12}) \tag{5.9}$$

After considerable symbolic computation, an approximate solution of the form

$$\psi \quad = \quad a\,(u,\,v,\,w)\;\exp\Big[-\frac{1}{2}\,(u\,+\,v\,+\,w)\Big] \tag{5.10}$$

is converted back to the $(r_1,\,r_2,\,r_{12})$ system, to compute electron density plots that are easily interpreted. The mechanized substitution of (5.9) into (5.10) gives

$$\psi \;=\; \bar a\,(r_1,\,r_2,\,r_{12})\;\exp\Big[-\frac{1}{2}\,\big(\epsilon\,(r_2 - r_1 + r_{12}) + \epsilon\;(r_1 - r_2 + r_{12}) + 2\,\epsilon\,(r_1 + r_2 - r_{12})\big)\Big] \tag{5.11}$$

Because $\bar a\,(u,\,v,\,w)$ is free of exponentials, this is converted to (5.12) by (5.13).

$$\psi \quad = \quad \bar a\,(r_1,\,r_2,\,r_{12})\;\exp\,(-\epsilon\,r_1 \;-\; \epsilon\,r_2) \tag{5.12}$$

$$\texttt{toThe[exp][ExpandAll]} \tag{5.13}$$

5·4 Multiple matches. To illustrate the usage of the `toEach` alias, we refer to the formation of (3.5) from the recurrence formula (5.14) for the Bessel functions $I_\nu(z)$ and the definition (5.15) of the $i_n(z)$.

$$I_{n+\frac{1}{2}}(z) \quad = \quad I_{n-\frac{3}{2}}(z) - \frac{(2n-1)}{z}\,I_{n-\frac{1}{2}}(z) \tag{5.14}$$

$$i_n(z) \quad = \quad z^{-(n+\frac{1}{2})}\sqrt{\frac{\pi}{2}}\,(2n+1)!!\,I_{n+\frac{1}{2}}(z) \tag{5.15}$$

These are recast as a recurrence relation between I_ν, $I_{\nu-1}$ and $I_{\nu-2}$, and an equation for I_ν in terms of $i_{\nu-\frac{1}{2}}$. A short sequence of mechanical operations leads to the representation of (5.16), containing the !! functions of the unexpanded arguments $2(\frac{n+1}{2})$, $2(\frac{n-1}{2})$ and $2(\frac{n-3}{2})$.

$$\sqrt{\frac{2}{\pi}}\,\frac{z^{n+\frac{1}{2}}}{\{2(n+\frac{1}{2})\}!!}\,i_n\,(z) \;=\; \sqrt{\frac{2}{\pi}}\,\frac{z^{n-\frac{3}{2}}}{\{2(n-\frac{3}{2})\}!!}\,i_{n-2}\,(z) - \sqrt{\frac{2}{\pi}}\,(2n-1)\;\frac{z^{n-\frac{3}{2}}}{\{2(n-\frac{1}{2})\}!!}\,i_{n-1}\,(z) \tag{5.16}$$

This is converted to the representation of (3.5) by

$$\texttt{toEach[bb][ExpandAll]} \tag{5.17}$$

5·5 Patterns as typifiers. To show the usage of a pattern as *typifier*, we refer to the skeletal form (5.18) of a long equation in [6].

$$\sum_{l,m,n=0}^{\infty} A_{l,m,n}\left[\ldots + \left(4\epsilon v^2 L_m(v)\, L_n(w) + \ldots\right)\left(-l\, L_l(u) + (u-1)\, L_l'(u)\right) + \ldots\right] = 0 \tag{5.18}$$

To allow the replacement of $uL_l'(u)$, using a recurrence formula, the sub-expression $(u-1)L_l'(u)$ must be expanded. The pattern object $(\_) * (u-1)$ matches any product of $(u-1)$. Accordingly, the targetting expression (5.19) was used, after we had found that $u-1$ occurs just once.

$$\texttt{toThe[(\_)*(u-1)][Expand]} \tag{5.19}$$

Another example occurs in an extension of the work in [3] that led to an expression of the form (5.20), where the $p_i(\tau)$ stand for uncollected polynomials in τ.

$$\begin{aligned}&\frac{1}{\tau^3}\Big[p_1(\tau) + p_2(\tau)\, e^{-2\tau} + p_3(\tau)\, e^{-(\kappa+2)\tau} + p_4(\tau)\, E_1[(\kappa+2)\tau] + p_5(\tau)\, e^{-2\tau} E_1[(\kappa-2)\tau] + \\ &p_6(\tau)\, e^{-2\tau} E_1(\kappa\tau) + p_7(\tau)\, e^{-2\tau} E_1[(\kappa+2)\tau] + p_8(\tau)\log\Big(\frac{\kappa}{\kappa-2}\Big) + p_9(\tau)\log\Big(\frac{\kappa+2}{\kappa}\Big)\Big]\end{aligned} \tag{5.20}$$

These polynomials are collected, *in situ*, by operating on the entire expression with

$$\texttt{toEach[(\_) + (\_\_)*tau}^{\texttt{n\_}\cdot}\texttt{]}\Big[\texttt{collectPowersOf[tau]}\Big] \tag{5.21}$$

The `(_)` is matched by one or several terms, and the `(__)` by many, one or no multipliers.

A pattern and an explicit sub-expression are used as *typifiers* in (5.22), which performs the conversion (4.20) by a variation of (4.22).

$$\texttt{inSuccession}\Big[\texttt{toThe[(\_)*pi][powerDown}^{2}\texttt{]},\ \texttt{toThe[2}^{\texttt{2j}}\texttt{][powerDown]}\Big] \tag{5.22}$$

5·6 A generalization. The `bilo` function that generalizes `toEachTermThat` has the aliases

$$\texttt{toEach}[\mathit{typifier}][\mathit{selector}][\mathit{action}][\mathit{target}] \tag{5.23a}$$
$$\texttt{toThe}[\mathit{typifier}][\mathit{selector}][\mathit{action}][\mathit{target}] \tag{5.23b}$$
$$\texttt{to}[\mathit{typifier}][\mathit{selector}][\mathit{action}][\mathit{target}] \tag{5.23c}$$

To illustrate the second of these, we refer to a step in the derivation of equation (2.5) of [4] that is abstracted by the conversion

$$\begin{aligned}&\sum_{n=0}^{\infty} a_n\, P_{n-1}(x) + \sum_{n=0}^{\infty} b_n\, P_n(x) + \sum_{n=0}^{\infty} c_n\, P_{n+1}(x) \quad \rightarrow \\ &\sum_{n=0}^{\infty} a_{n+1}\, P_n(x) + \sum_{n=0}^{\infty} b_n\, P_n(x) + \sum_{n=0}^{\infty} c_{n-1}\, P_n(x)\end{aligned} \tag{5.24}$$

where $a_0 = c_{-1} = 0$. We represent $\sum_{i=j}^{k} f$ by

$$\texttt{sum[i, j, k][f]} \tag{5.25}$$

and perform the transformation by (5.26), where some `bilo` functions mentioned in [8] reindex the series and, respectively, expand out $a_0 P_{-1}(x)$ from the first reindexed series and subtract $c_{-1} P_0(x)$ to compensate for its inclusion in the third.

```
inSuccession[toThe[sum][that[contains[a]]][reindex[n,n-1], leftExpand]
             toThe[sum][that[contains[c]]][reindex[n,n+1], leftExtend]]     (5.26)
```

A pointer or pointer list may begin or be the entirety of *selector*, as when dealing with (5.27), which occurs in [5].

$$\sum_{n=0}^{\infty}\sum_{m=0}^{\infty}\sum_{l=0}^{\infty} a_{l,m,n}\, L_l(u) \quad\rightarrow\quad \sum_{l=0}^{\infty} L_l(u) \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} a_{l,m,n} \tag{5.27}$$

The starting expression is represented by

```
sum[n, 0, infinity][sum[m, 0, infinity][sum[l, 0, infinity][a[l, m, n] * L[l, u]]]]  (5.28)
```

The conversion is performed by (5.29), using two more `bilo` functions with self-explanatory names.

```
inSuccession[to[sum][2][switchOrderOfSummation],
             to[sum][3][moveOutTheIndependentFactor],
             to[sum][1][switchOrderOfSummation],
             to[sum][2][moveOutTheIndependentFactor]]                       (5.29)
```

5·7 Disambiguation. In (5.23a) and (5.23b), *selector* must begin with (or comprise entirely) a pointer, a pointer list, or `that[`*selector'*`]`, where *selector'* usually is a criterion, such as `contains[x]`, but can take any form described for *selector* at the end of Section 4. Also, in the `toEach` and `toThe` functions, *action* cannot begin with (or comprise) an expression with the head `that`. This is not bothersome in practice. These restrictions disambiguate (5.1) from the first two forms in (5.23). The functions `toThe[`*typifier*`][`*action*`]` and `toEach[`*typifier*`][`*action*`]` evaluate to (5.30). The implementation of this function in *Mathematica* primitives is 9 lines long, supplemented by a 10 line procedure to accommodate rules and exponents in *action*.

`to[`*typifier*`][ ][`*action*`][`*target*`]` (5.30)

5·8 Extensibility. We define wrappers for frequently used functions typified by

`toEachLog, toTheLog` $\Rightarrow$ `to[log][ ]` (5.31)

`toEachLogThat, toTheLogsThat, toTheLogThat, toLogs, toLog` $\Rightarrow$ `to[log]` (5.32)

The `bilo` procedures that define these functions are constructed mechanically from lists of the roots, (such as `Log, Exp, Sum, Product, Derivative`) and the corresponding heads that are used in the `bilo` representations (`log, exp, sum, prod, D$` for the examples just cited).

6. TARGETTING THE ARGUMENTS AND SUB-HEADS OF A FUNCTION

6·1 Basic constructions. The largest class of `bilo` addressing functions that have been used so far is based on (6.1). Several further functions are based on (6.2). These act on selected arguments of the head of the target expression, individually in the case of (6.1) and collectively in the case of (6.2).

`toArguments[`*selector*`][`*action*`][`*target*`]` (6.1)

`collectivelyToArguments[`*selector*`][`*action*`][`*target*`]` (6.2)

The immediate wrappers of (6.1) analogous to (5.1) and (5.23) are, respectively,

`toEachArgument, toTheArgument` $\Rightarrow$ `toArguments[ ]` (6.3)

`toEachArgumentThat, toTheArgumentsThat, toTheArgumentThat, toArgument`
$\Rightarrow$ `toArguments` (6.4)

Also, (6.5), and its counterparts with 2 and 3 in place of 1, are used.

`toArgument1` $\Rightarrow$ `toArgument[1]` (6.5)

The usage of `toTheArgument` is illustrated by the conversion (6.6) which begins with the result (5.12) obtained earlier. It is effected by (6.7).

$$\psi = \bar{a}\,(r_1,\, r_2,\, r_{12})\ \exp\,[-\epsilon\, r_1\, - \epsilon\, r_2] \qquad \to \qquad \psi = \bar{a}\,(r_1,\, r_2,\, r_{12})\ \exp\,[-\epsilon\,(r_1\, +\, r_2)] \tag{6.6}$$

$$\texttt{toThe[exp]}\big[\,\texttt{toTheArgument[Factor]}\,\big] \tag{6.7}$$

Another example occurs in the derivation of equation (5.6) of [4]. This involves a conversion of the form (6.8), where a, b and c stand for long sub-expressions that do not contain sin.

$$a \sum_{k=0}^{\infty} \Big[\, b\ \sin\big(\frac{(2j\, -\, 2k)\,\theta}{2}\big)\ +\ c\ \sin\big(\frac{(2j\, -\, 2k)\,(\theta\, +\, \pi)}{2}\big)\,\Big] \qquad \to$$

$$a \sum_{k=0}^{\infty} \big[\, b\ \sin\big((j\, -\, k)\,\theta\big)\ +\ c\ \sin\big((j\, -\, k)\,\theta\, +\, (j\, -\, k)\,\pi\big)\,\big] \tag{6.8}$$

The conversion is performed by (6.9), using the `bilo` function `leftDistribution` (see (3.6)).

$$\begin{aligned}&\texttt{inSuccession}\Big[\texttt{toThe[sin]}\,\Big[\texttt{that}\big[\texttt{doesNotContain[pi]}\big]\Big]\,\big[\texttt{toTheArgument[Factor]}\big],\\ &\quad \texttt{toThe[sin]}\,\Big[\texttt{that}\big[\texttt{contains[pi]}\big]\Big]\,\big[\texttt{toTheArgument[Factor, leftDistribution]}\big]\Big]\end{aligned} \tag{6.9}$$

6·2 Secondary constructions. When f is a unary function, `toEachArgument` acts as the built-in `Map` (see [2], p.204).

$$g\texttt{[}x_1,\, x_2,\, \dots\,,\, x_n\texttt{]}\ \texttt{// toArguments[}f\texttt{]} \quad \Rightarrow \quad g\big[f\texttt{[}x_1\texttt{]},\, f\texttt{[}x_2\texttt{]},\, \dots\,,\, f\texttt{[}x_n\texttt{]}\big] \tag{6.10}$$

In general, `inSuccession[`*action*`]` is applied to each x_i by `toEachArgument[`*action*`]`, and to just the selected x_i by `toEachArgument[`*selector*`][`*action*`]`.

Many addressing functions of `bilo` are implemented as nested expressions of the form (6.11), and as shorter forms obtained by combining `toEach` and `toThe` with the aliases (6.3)–(6.5).

$$\texttt{to[}\mathit{typifier}\texttt{][}\mathit{selector}\texttt{]}\,\big[\texttt{toArguments[}\mathit{selector}'\texttt{][}\mathit{action}\texttt{]}\big] \tag{6.11}$$

Thus, because the full forms of $a == b$, $a >= b$, ... are `Equal[`a, b`]`, `GreaterEqual[`a, b`]`, ..., in *Mathematica*, the functions (3.4) are implemented by

$$\texttt{relHeads}\ \ =\ \ \texttt{Equal|GreaterEqual|Greater|...} \tag{6.12}$$

$$\begin{aligned}&\texttt{toTheLhs[}\mathit{action}\texttt{]} && \Rightarrow\ \texttt{toThe[relHeads]}\big[\texttt{toArguments][1][}\mathit{action}\texttt{]}\big]\\ &\texttt{toTheRhs[}\mathit{action}\texttt{]} && \Rightarrow\ \texttt{toThe[relHeads]}\big[\texttt{toArguments][2][}\mathit{action}\texttt{]}\big]\\ &\texttt{toBothSides[}\mathit{action}\texttt{]} && \Rightarrow\ \texttt{toThe[relHeads]}\big[\texttt{toArguments][\{1,2\}][}\mathit{action}\texttt{]}\big]\end{aligned} \tag{6.13}$$

Also, because *Mathematica* represents `(a + b + ... )` and `(a × b × ... )` as `Plus[a, b, ... ]` and `Times[a, b, ... ]`, respectively, the functions of Section 4 are implemented by

$$\begin{aligned}&\texttt{toEachTermThat[}\mathit{selector}\texttt{][}\mathit{action}\texttt{]}\\ &\Rightarrow\quad \texttt{to[Plus][outermost]}\big[\texttt{toArguments[}\mathit{selector}\texttt{][}\mathit{action}\texttt{]}\big]\\ &\texttt{toEachFactorThat[}\mathit{selector}\texttt{][}\mathit{action}\texttt{]}\\ &\Rightarrow\quad \texttt{to[Times][outermost]}\big[\texttt{toArguments[}\mathit{selector}\texttt{][}\mathit{action}\texttt{]}\big]\end{aligned} \tag{6.14}$$

$$\begin{aligned}&\texttt{toEachTerm[}\mathit{action}\texttt{]}\\ &\Rightarrow\quad \texttt{to[Plus][outermost]}\big[\texttt{toArguments[ ][}\mathit{action}\texttt{]}\big]\\ &\texttt{toEachFactor[}\mathit{action}\texttt{]}\\ &\Rightarrow\quad \texttt{to[Times][outermost]}\big[\texttt{toArguments[ ][}\mathit{action}\texttt{]}\big]\end{aligned} \tag{6.15}$$

This lets the functions that act on terms and factors reach the outermost Plus and Times sub-expressions in the target, without affecting inner sub-expressions of these kinds, too. Thus,

$$\begin{aligned}&\Big(\texttt{a}*\big(\texttt{b}+\texttt{c}*\texttt{(d+e)}\big)\ \ ==\ \ \texttt{f[x]}+\texttt{g[a+h[b+c]]}\Big)\ \texttt{// toEachTerm[F]}\\ &\Rightarrow\quad \texttt{a}*\big(\texttt{F[b]}+\texttt{F}\big[\texttt{c}*\texttt{(d+e)}\big]\big)\ \ ==\ \ \texttt{F}\big[\texttt{f[x]}\big]+\texttt{F}\big[\texttt{g[a+h[b+c]]}\big]\end{aligned} \tag{6.16}$$

At times, a mnemonic head is put on a sequence of items, to allow the use of further mnemonic addressing functions. Thus, denoting a vector by $\texttt{vector[}v_1,\ v_2,\ \ldots\texttt{]}$ permits

$$\begin{aligned}&\texttt{toVectorElements[}selector\texttt{][}action\texttt{]}\\ &\Rightarrow\quad \texttt{toThe[vector][toArguments][}selector\texttt{][}action\texttt{]]}\end{aligned}\tag{6.17}$$

We define wrappers for many more functions covered by (6.10) and its shorter forms. For example, (6.18) acts on the summands of the selected sums.

$$\texttt{toEachSummandThat[}selector\texttt{][}action\texttt{]}\ \Rightarrow\ \texttt{to[sum][}selector\texttt{]}\big[\texttt{toTheArgument[}action\texttt{]}\big]\tag{6.18}$$

This is an instance of a set of aliases analogous to (5.23) that contain a word connoting an argument, which make the selection amongst the sub-expressions with the heads that the word suggests (e.g. `Power` for `Base` and `Exponent`, trigonometric functions for `Angle`). Such functions will be defined in accounts of the applications.

Because of the way in which *Mathematica* represents fractions, the built-in `Numerator` and `Denominator` functions are used to define the `bilo` functions that address these, as in

$$\texttt{toTheNumerator[}action\texttt{][}s\texttt{]}\Big]\quad\Rightarrow\quad\frac{\texttt{inSuccession[}action\texttt{]}\big[\texttt{Numerator[}s\texttt{]}\big]}{\texttt{Denominator[}s\texttt{]}}\tag{6.19}$$

6·3 Some extensions. At times, an entire sequence of arguments must be treated as an entity. Several `bilo` functions that deal with argument lists just use simple combinations of elementary list operations of *Mathematica*, (see [2], Section 1.8). Thus, (6.20) swaps arguments at positions $q_1,\ q_2 > 0$. A slightly longer expression lets $-q$ point to the q'th item counting from the right.

$$\begin{aligned}&\texttt{swapArguments[}q_1,\ q_2\texttt{][}s\texttt{]}\quad\Rightarrow\\ &\texttt{s}\Big[\Big[\texttt{Join}\Big[\texttt{Range[1,}\ q_1\texttt{-1]},\ \{q_2\},\ \texttt{Range[}q_1\texttt{+1,}\ q_2\texttt{-1]},\ \{q_1\},\ \texttt{Range}\big[q_2\texttt{+1, Length[s]}\big]\Big]\Big]\Big]\end{aligned}\tag{6.20}$$

The functions `collectivelyToTheTerms` and `collectivelyToTheFactors`, discussed in Section 4, simply wrap the more general function which has the aliases `collectivelyToArguments` and `collectivelyToTheArguments`.

$$\begin{aligned}&f\texttt{[}x_1,\ x_2,\ \ldots\texttt{]}\ \texttt{//}\ \texttt{collectivelyToArguments[}selector\texttt{][}action\texttt{]}\\ &\Rightarrow\quad f\big[\texttt{inSuccession[}action\texttt{][}x_{k_1},\ x_{k_2},\ \ldots\texttt{]},\ x_{l_1},\ x_{l_2},\ \ldots\big]\end{aligned}\tag{6.21}$$

where $x_{k_1},\ x_{k_2},\ \ldots$ are the arguments picked by *selector* and $x_{l_1},\ x_{l_2}$ are the arguments which are not. This function, in turn, wraps `toTheArgumentList` where, when *action* is a single head,

$$f\texttt{[}x_1,\ \ldots,\ x_n\texttt{]}\ \texttt{//}\ \texttt{toTheArgumentList[}action\texttt{]}\quad\Rightarrow\quad f\big[action\texttt{[}\{x_1,\ \ldots,\ x_n\}\texttt{]}\big]\tag{6.22}$$

When *action* is a sequence of heads and/or rules, $\texttt{inSuccession[}action\texttt{]}$ appears on the right hand side, in place of *action*. Also, when $0 < q_1 < q_2 \leq n$,

$$\begin{aligned}&f\texttt{[}x_1,\ \ldots,\ x_n\texttt{]}\ \texttt{//}\ \texttt{collectivelyToConsecutiveArguments[}q_1,\ q_2\texttt{][}F\texttt{]}\\ &\Rightarrow\quad f\big[x_1,\ \ldots,\ x_{q_1-1},\ F\texttt{[}x_{q_1},\ \ldots,\ x_{q_2}\texttt{]},\ x_{q_2+1},\ \ldots,\ x_n\big]\end{aligned}\tag{6.23}$$

When q_1 and/or q_2 is negative, the unsigned value counts from the right.

The functions defined by (6.21)–(6.23) are used in conjunction with elementary list operations of *Mathematica*, and with further `bilo` functions that partition lists. These include

$$\{v_1,\ \ldots,\ v_K\}\ \texttt{//}\ \texttt{partInTwo[}selector\texttt{]}\ \Rightarrow\ \big\{\{v_{\mu_1},\ \ldots,\ v_{\mu_M}\},\ \{v_{\nu_1},\ \ldots,\ v_{\nu_N}\}\big\}\tag{6.24}$$

where the $v_{\mu_1}, \ldots v_{\mu_M}$ are picked from the $v_1, \ldots v_K$ by *selector* and the $v_{\nu_1}, \ldots v_{\nu_N}$ are the v_k which are not. The original order of the elements is preserved in the two sub-lists. In the more general function (6.25), s_ν is the list of x_i for which $c_1(x),\ \ldots,\ c_{\nu-1}(x)$ is false and $c_\nu(x)$ is true, for s_1 to s_m, and s_{m+1} is the list of x_i for which $c_1(x),\ \ldots,\ c_m(x)$ are false.

$$\{x_1,\ \ldots,\ x_n\}\ \texttt{//}\ \texttt{partByCriteria[}c_1,\ \ldots,c_m\texttt{]}\quad\Rightarrow\quad\{s_1,\ \ldots,\ s_{m+1}\}\tag{6.25}$$

Further `bilo` functions direct action to the elements of the compound heads that we use to represent functionals such as indexed sums and products, integrals, derivatives and limits.

7. TARGETTING ON COEFFICIENTS

7·1 A prototype. Often, it is necessary to operate on the coefficient of a unique sub-expression, or on the coefficients in a polynomial or in a Plus of sub-expressions which have some other systematic structure, such as terms containing $f(x_1)$, $f(x_2)$, ... for some function f. Earlier sections contain several examples of such situations. `bilo` contains a few basic addressing functions that deal explicitly with coefficients, and can be wrapped in specialized functions for particular applications. To illustrate the usage of the first of the basic "coefficient" functions, we refer to the reduction (7.1), where $\Delta = \sqrt{1 - k^2 \sin^2 x}$ (see [11], Section 2.584).

$$\int \frac{\cos^3 x\, dx}{\Delta^5} \quad\rightarrow\quad \int \frac{\cos x\, dx}{\Delta^5} - \int \frac{\sin^2 x \, \cos x\, dx}{\Delta^5} \quad\rightarrow\quad \frac{(3\;-\;2\,k^2 \sin^2 x)\, \sin x}{3\,\Delta^3} - \frac{\sin^3 x}{3\,\Delta^3} \quad\rightarrow$$
$$\frac{\big(3\;+\;(-1\;-\;2\,k^2)\,\sin^2 x\big)\,\sin x}{3\,\Delta^3} \quad\rightarrow\quad \frac{\big(3\;-\;(1\;+\;2\,k^2)\,\sin^2 x\big)\,\sin x}{3\,\Delta^3} \tag{7.1}$$

The first few steps are handled by simple *Mathematica* functions, and the final step by

$$\texttt{toTheCoefficientOf}\big[\texttt{sin[x]}^2\big]\big[\texttt{factorOut[-1]}\big] \tag{7.2}$$

This contains an addressing function that seeks coefficients of the specified item throughout the target expression. It has the aliases

`toTheCoefficientOfThe`[*typifier*][*action*] , `toEachCoefficientOf`[*typifier*][*action*] ,
`toTheCoefficientOfEach`[*typifier*][*action*] , `toTheCoefficientOf`[*typifier*][*action*] (7.3)

7·2 Another example. The mechanical reduction of the integrals typified by the J_{13} of (4.9) leads initially to expressions containing $\kappa = \alpha/\beta$ and $\tau = \beta\,\rho$. Thus,

$$J_1 \;=\; \int e^{-\alpha\, r_a \,-\, \beta\, r_b}\, dV \;=\; \frac{8\pi}{\beta^3\,(\kappa^2-1)^3\,\tau}\big[(\kappa^3\,\tau - \kappa\,\tau - 4\,\kappa)\,e^{-\tau} \;+\; (\kappa^2\,\tau - \tau + 4)\,e^{-\kappa\,\tau}\big] \tag{7.4}$$

This is reduced to (7.5) by (7.6). Any of the aliases in (7.3) could be used. We chose the one that seemed most natural.

$$\frac{8\pi}{\beta^3\,(\kappa^2-1)^3\,\tau}\Big[\big(\tau\,(\kappa^3\;-\;\kappa) - 4\,\kappa\big)\,e^{-\tau} \;+\; \big(\tau\,(\kappa^2\;-\;1)\;+\;4)\big)\,e^{-\kappa\,\tau}\Big] \tag{7.5}$$

$$\texttt{toTheCoefficientOfEach[exp]}\,\big[\,\texttt{collectPowersOf[tau]}\,\big] \tag{7.6}$$

Then this is reduced to (7.7) by (7.8). Another of the aliases in (7.3) is used here, again for naturalness of expression.

$$\frac{8\pi}{\beta^3\,(\kappa^2-1)^3\,\tau}\Big[\big(\tau\,\kappa\,(\kappa\;-\;1)\,(\kappa\;+\;1)\;-\;4\,\kappa\big)\,e^{-\tau} \;+\; \big(\tau\,(\kappa\;-\;1)\,(\kappa\;+\;1)\;+\;4\big)\,e^{-\kappa\,\tau}\Big] \tag{7.7}$$

$$\texttt{toEachCoefficientOf[tau][Factor]} \tag{7.8}$$

7·3 Using patterns. The $(\kappa^2-1)^3$ is not recognized as part of the coefficient of an occurrence of τ and is not factored by (7.8), because *Mathematica* represents the τ in the denominator as τ^{-1}. The pattern $\tau^{-.}$ matches τ^n for **any** $n \neq 0$ (see [2], p.234). So, using (7.9), $(\kappa^2-1)^3$ is factored, too.

$$\texttt{toTheCoefficientOfEach[tau}^{\texttt{n\_.}}\texttt{]\,[factorIfPossible]} \tag{7.9}$$

The `bilo` function `factorIfPossible` factors a `Plus` that is not prime and leaves it unaltered, otherwise. In contrast, the built-in `Factor` leaves a prime argument expanded. Also, `factorIf-Possible` acts separately on the arguments of a `Times`. It is used in (7.9) because `Factor` would expand the expression in square brackets in (7.5).

Another example of patterns used to identify coefficients occurs in [5], when dealing with equations of the form (7.10). The $c_{n-2}, \ldots, c_{n+2}$ are Plus expressions that consist of hundreds of terms containing Laguerre polynomials of u and v, but independent of w.

$$\sum_{l=0}^{\infty}\sum_{m=0}^{\infty}\sum_{n=0}^{\infty}\left[c_{n-2}\,L_{n-2}(w) \;+\; c_{n-1}\,L_{n-1}(w) \;+\; \ldots \;+\; c_{n+2}\,L_{n+2}(w)\right] \quad = \quad 0 \qquad (7.10)$$

We operate systematically on the c coefficients by using (7.11). The pattern object here matches any Laguerre polynomial of w, but not of u or v.

$$\texttt{toTheCoefficientOfEach}\big[\texttt{L[\_, w]}\big]\texttt{[}\mathit{action}\texttt{]} \qquad (7.11)$$

7·5 Targetting a Plus. The processes that reduce J_1 to (7.4) give an analogous expression for J_{13}, containing terms in τ^2 as well as τ. This is reduced to (7.12) by (7.13).

$$\begin{aligned}\frac{8\pi}{\beta^5\,(\kappa^2-1)^5\,\tau} \times \Big[\big\{ & 20 \,+\, 152\,\kappa^2 \,+\, 20\,\kappa^4 \,-\, 8\,\tau\,(\kappa-1)\,(\kappa+1)\,(5\,\kappa^2+1) + \\ & \tau^2\,(\kappa-1)^2\,(\kappa+1)^2\,(3\,\kappa^2+1) \,\big\}\, e^{-\tau} + \\ \big\{ & -20 \,-\, 152\,\kappa^2 \,-\, 20\,\kappa^4 \,-\, 8\,\tau\,\kappa\,(\kappa-1)\,(\kappa+1)\,(\kappa^2+5) - \\ & \tau^2\,(\kappa-1)^2\,(\kappa+1)^2\,(\kappa^2+3) \,\big\}\, e^{-\kappa\tau} \Big]\end{aligned} \qquad (7.12)$$

```
toTheCoefficientOfEach[exp] [ collectPowersOf[tau] ,
                              toTheCoefficientOfEach[tau^n_.][Factor] ,
                              toEach[-kappa + _] [factorOut[-1]] ]        (7.13)
```

A more useful result is obtained by also factoring the part of the coefficient of each exp that is independent of τ, to give,

$$\frac{8\pi}{\beta^5\,(\kappa^2-1)^5\,\tau}\Big[\big\{4(5+38\,\kappa^2+5\,\kappa^4) \,+\, \ldots\big\}\,e^{-\tau} \,+\, \big\{-\,4(5+38\,\kappa^2+5\,\kappa^4) \,+\, \ldots\big\}\,e^{-\kappa\tau}\Big] \qquad (7.14)$$

We deal with this and similar situations using the bilo function

$$\texttt{toTheCoefficientsOfPowersOf}\,\texttt{[}v\texttt{][}\mathit{action}\texttt{]} \qquad (7.15)$$

This treats the target expression as a polynomial in v and applies inSuccession[*action*] to each of the coefficients that characterizes the polynomial, including the implicit coefficient of v^0, that is, the Plus of the independent terms. The expression (7.14) is formed from the precursor of (7.12), accordingly, by

```
toTheCoefficientOfEach[exp] [ collectPowersOf[tau] ,
                              toTheCoefficientsOfPowersOf[tau][Factor] ,
                              toEach[-kappa + _] [factorOut[-1]] ]        (7.16)
```

The function (7.15) wraps the primitive

$$\texttt{toTheCoefficientsInTheTargetPlusOf}\,\texttt{[}\mathit{typifier}\texttt{][}\mathit{action}\texttt{]} \qquad (7.17)$$

This applies inSuccession[*action*] only to the coefficient part of terms that contain *typifier* as a factor (or which consist entirely of *typifier*, with 1 as the implied coefficient) and to the Plus of the other terms. In contrast, the function (7.3) acts on occurrences of *typifier* at all levels of the target expression. The function(7.17) has further wrappers that act on coefficients of sines and cosines of a given argument and on arbitrary functions of the argument. Also, the function

$$\texttt{toTheProperCoefficientsInTheTargetPlusOf}\,\texttt{[}\mathit{typifier}\texttt{][}\mathit{action}\texttt{]} \qquad (7.18)$$

acts on the same coefficients as (7.17) except for the Plus of the independent terms.

7·5 Factors. Each bilo function that addresses a coefficient acts on its entirety. At times, it is necessary to operate on individual factors. Consider, for example, (7.19), where each a_i is independent of a parameter ϵ and each b_i is an uncollected polynomial in ϵ, that is to be collected.

$$a_1\,b_1\,f(x_1) \;+\; a_2\,b_2\,f(x_2) \;+\; \ldots \qquad (7.19)$$

The collect functions act only on Plus expressions. Accordingly, (7.20) is used

$$\texttt{toTheCoefficientOfEach[f]}\Big[\ \texttt{toTheFactorThat}\big[\texttt{contains[epsilon]}\big]\ \big[\texttt{collectPowersOf[epsilon]}\big]\Big] \tag{7.20}$$

8. FORMATTING THE OUTPUT

8·1 Basics. The arrangement of output is extremely important when using symbolic computation for instruction and publication. Several bilo functions format and skeletalize. These

1. suppress commutativity of $+$ and $-$ by clearing the attribute Orderless of Plus and Times,
2. rearrange the order of terms and factors, in accordance with criteria that the user specifies,
3. replace sequences of terms, factors, elements of vectors, rows and columns of matrices and so forth by dots or $\ll\ \gg$ expressions, when required as in (8.3) below,
4. apply the *Mathematica* function HoldForm to the relevant expressions, to insulate these from reordering when commutativity is restored, (see [2], p.281),
5. restore the commutativity for use in subsequent operations.

Mathematica suppresses the keyword HoldForm and the associated brackets in the output.

A simple example is provided by assigning n(n+1)(n+2)/6 to a variable s and then typing s. This produces the display (8.1) in the default style of *Mathematica*.

$$\frac{\mathtt{n\,(1+n)\,(2+n)}}{6} \tag{8.1}$$

The bilo function numbersLast moves numerical terms in a Plus to the end, and puts HoldForm on the result. Consequently, typing (8.2a) produces the display (8.2b).

s // toEachFactor[numbersLast] (8.2a), $\qquad \dfrac{\mathtt{n\,(n+1)\,(n+2)}}{6}$ (8.2b)

8·2 A longer example. Nesting is used extensively, to target the formatting and skeletalization of very long expressions. An intermediate result in a major derivation in [5] is displayed, in effect, as (8.3), with the sum represented by sum[{l, m, n}, 0, infinity][...], the Laguerre functions as L[m, x] and the skeletal expressions as <<1 term>>, <<5 terms>> and so forth.

$$\sum_{l,m,n=0}^{\infty}\Big[A_{l,m,n}\Big\{\big[l\,(l-1)\,L_{l-2}(u)\ -\ 2\,l\,(l-1)\,L_{l-1}(u)\ +\ l\,(l-1)\,L_l(u)\big]\times \big[4\,\epsilon\,v^2\,L_m(v)\,L_n(w)+\ <<1\ \text{term}>>\big] + \ <<5\ \text{terms}>>\ + L_l(u)\,\big[-4\,\epsilon\,v^2\,L_m(v)\,L_n(w)+\ <<18\ \text{terms}>>\big]\Big\}\Big] \ = \ 0 \tag{8.3}$$

The default output of the term shown as $l\,(l-1)\,L_{l-2}(u)$ is (-1+l) l L[-2+l, u]. The expression (8.4) produces a close depiction of (8.3) from the full equation. The sortByAbsence[c_1, ..., c_n] function pulls out the factors that do not contain any of the $c_1, \ldots, c_n$, retaining the order that they had in the target expression, then prepends the residual factors that do not contain any of the $c_2, \ldots, c_n$, continuing in like manner to the residual factors that do not contain c_n, and then to those that do. For the formatting that is useful in work considered here, this way of specifying the sequencing requirements is much more convenient than the pairwise comparison required by the built-in Sort function (see[2], p.130). The sortByAbsence and related sorting functions use the toTheArgumentList and partByCriteria functions mentioned in Section 6.

Here, sortByAbsence[Plus, L] pulls out the factor l, then $(l-1)$ and then the Laguerre function. allowFurtherSorting suppresses commutativity of $+$ and $\times$ and removes the HoldForm imposed earlier. The disallowFurtherSorting restores HoldForm and commutativity.

```
toTheSummand[toTheFactorThat[contains[u]]
  [showTerms[{1,-1}], allowFurtherSorting,
   toTerm[1][[toTheFactorThat[contains[u]][toEachTerm[[sortByAbsence[Plus,L]]]]],
   toTerms[{1,-1}][toTheFactorThat[contains[v]]][showTerm[{1}]]],
   numbersLast, disallowFurtherSorting]                                   (8.4)
```

8·3 Patterns. Sometimes, quite short tactics work. A recurrence formula in [5] contains 60 sub-expressions that need to have the numbers at the right, and all of these consist of l, m or n plus or minus an integer. The pattern in (8.5) matches these but no other `Plus` expression in the equation. Accordingly, this part of the formatting is done by applying (8.5) to the entire equation.

```
to[_Integer+_Symbol][numbersLast]                                         (8.5)
```

9. DISCUSSSION

9·1 Some benefits. This methodology was developed in part because MPB has had to repeat long manual derivations, with minor variations, several times over a period of forty years, and he wanted to capture further tedious derivations in a way that could be modified mechanically. Resources to perform rederivations by machine accommodate many working situations that include:

1. minor changes of notation,
2. changes in the sign conventions and in the normalization and other scaling factors in the definitions of special functions,
3. variations in the amount of detail that is printed,
4. changing the steps in parts of a derivation,
5. incorporation of derivations in new expository contexts.

It is important that the operations which lead from step to step are easy to understand. Function names were chosen, accordingly, to allow `bilo` statements of the form (9.1) to be verbalized as (9.2), in reasonable conformity to natural language usage.

$$v \;//\; \texttt{inSuccession[}f_1,\ f_2,\ \ldots\texttt{]} \tag{9.1}$$

$$\text{take } v \text{ and apply in succession } f_1,\ f_2,\ \ldots \tag{9.2}$$

One major feature of our work is the ability to focus action precisely. Other workers also stress the need for precise control of the structure of expressions produced by symbolic computation. For example, the ability to meet this need by cursor control is adduced as a major advantage of the 'Theorist' system [13], which may be more convenient for some purposes than the `bilo` approach. Detailed comparisons on a variety of applications would be useful.

The development of `bilo` has progressed in conjunction with that of `autorun`, a front end interpreter reported in part in [7]. This reads control files that consist of

1. executable *Mathematica* statements that are part of a derivation,
2. lines of text that are passed to the output,
3. *Mathematica* statements that control output styles and format,

These are identified and distinguished in simple ways. A collection of procedures to produce the output in TeX is well in hand. These depend heavily on the methods of this paper.

9·2 Coding. The early versions of `bilo` evolved in conjunction with the applications [4–6] over many months of irksome coding, hampered by bugs in Release 1.2 of *Mathematica*. Reimplementation using Release 2 is proceeding rapidly, and the new `bilo` procedures are very short. They use `Apply`, `Composition`, `FixedPoint`, `Map`, `MapAt`, `Nest`, `Part`, `Position`, `ReplaceAll`, `ReplaceRepeated`, `Select` and `Through`, and can be explained quite concisely. These built-in functions could be used directly. However, `bilo` derivations can be read by users of applied mathematics who are not yet motivated to read derivations that make explicit use of the *Mathematica* functions just mentioned.

The `factorOut[-1]` function, used several times here, highlights the constrast that can occur between mathematical triviality of an operation, and the programming knowledge needed to code it. The `UpSetDelayed` function that is used, (see [2], p.260), is not one of the first that a casual user of *Mathematica*, preoccupied with the subject matter of an application, might consider. The problem results from the unsuppressable conversion of $-(a+b)$ to $-a-b$ by *Mathematica*. Just as commutativity of `Times` and `Plus` can be switched on and off, software switches to enable and disable various features of automatic evaluation would be helpful in symbolic calculation systems.

Because so much can be done with so few *Mathematica* functions, it may be worth studying the reimplementation of the `bilo` functions in a general purpose programming language. In effect, every addressing function in this paper operates on a tree with labelled nodes, by seeking either

1. a terminal sub-tree or a class of terminal sub-trees,
2. a labelled node or a class of labelled nodes,

by reference to an explicit reference or a pattern that these match, then operating on either

1. the terminal sub-trees picked in case 1 or headed by the nodes picked in case 2,
2. the immediate sub-trees in case 2 (i.e. the arguments),
3. the siblings (i.e. the coefficients).

Operations of these kinds are relevant to appropriate representations of information of many kinds besides mathematical expressions.

9·3 Further work. To deal with the problems of elementary algebraic manipulation, a start has been made on functions, analogous to the present `bilo` hierarchical addressing scheme, to perform construction and extraction instead of composition and *in situ* conversion. Also, notations are being devised to represent assertions and definitions, such as occur in modern algebra, in the hierarchical style with which `bilo` deals. A starting point was provided by the conventions for conditional equations and associated rules in a quasi-axiomatic derivation of elementary trigonometric formulas from geometrical considerations, and by ways to reduce integrals of series expansions in special functions by use of orthogonality.

The efforts with respect to all these matters, like the substance of [8] and the present paper, are consequent on encountering real practical needs in the course of varied and extensive applications. Above all else, we emphasize our perception of symbolic computation as an area of experimentation, in which the goal of doing by machine what can be done by hand, as well as what cannot, must be considered.

References

1. C.B.Boyer, *A history of mathematics*, Wiley, New York, NY (1968), (p.252 discusses Arabic terms translated as "restoration" and "completion" in Robert of Chester's *Latin Translation of the algebra of al-Khowarizmi*, ed. L.C.Karpinski, as operations on the sides of an equation).
2. S.Wolfram, *Mathematica, A system for doing mathematics by computer, 2nd edition*, Addison-Wesley, New York, NY, 1991.
3. M.P.Barnett, *Molecular integrals over slater orbitals*, Chem. Phys. Letters **166** (1990), 65-70.
4. M.P.Barnett, *Summing $P_n(\cos\theta)/p(n)$ for certain polynomials $p(n)$*, Computers Math. Applic. **21** (1991), 79-86.
5. M.P.Barnett, *Symbolic calculation of Pekeris wave functions*, to be published.
6. Z-z.Chen, *Symbolic calculation of inverse kinematics of robot manipulators*, M.A. thesis, Brooklyn College of the City University of New York, 1991.
7. M.P.Barnett, *Some simple ways to construct and to use formulas mechanically*, ACM SIGSAM Bulletin **28** **(2)** (1991), 21-29.
8. M.P.Barnett, *Implicit rule formation in symbolic computation*, Computers Math. Applic., in press.
9. J.R.Hindley and J.P.Seldin, *Introduction to combinators and lambda-calculus*, Cambridge University Press, New York, NY, 1986.
10. C.A.Coulson, *Two-centre integrals occurring in the theory of molecular structure*, Proc. Camb. Phil. Soc. **38** (1042), 210-232.
11. I.S.Gradshtyn and I.M.Rhyzhik, *Table of integrals, series and products,* [tr. and ed. by A.Jeffrey], Academic Press, New York, NY, 1980.
12. C.L.Pekeris, *Ground state of two-electron atoms*, Phys. Rev. **112** (1958), 1649-1658.
13. J.Rizzo, *Theorist*, MacUser (June, 1990), 57-59.