

# SMP Language Summary (Feb 7, 1980)

## Data Types:

### 1. Numbers (Numb)

Floating-point 64 bit numbers with or without 10 error estimates.

### 2. Symbols (Symb)

Symbols are specified by their print names, which consist of arbitrary-length strings (if containing non-alphanumerics, enclosed in " "). Symbols may be assigned values and properties. Function names are symbols. Extended symbols consist of ordinary symbols multiplied by signed rational coefficients (between  $2^{-31}$  and  $2^{31}$ ). Rational numbers are therefore considered as special extended symbols.\* Ordinary symbols whose print names begin with \$ will be considered as dummy (or arbitrary) symbols. Internally-generated symbols will have names beginning with #.

Both user- and system-defined functions are specified by their names, which are ordinary symbols. These symbols may be assigned properties (which define the type of expression "returned" by the function) and "values" (which specify patterns for substitution). System-defined functions have in addition a text routine which performs simplification and evaluation. Special system-defined functions may be entered and are

\* Arbitrary-length integers are considered and specified as concatenations of ordinary integers.

output in a non-prefix notation.

### 3. Expressions (Expr)

Expressions consist of combinations of symbols (and perhaps numbers). Internally, expressions contain the additional data type Limbs, which serve to connect and delimit the trees which represent the expressions.

#### Examples

Numbers :  $3.14$ ,  $7.95 + 0(0.55)$ ,  $6.6e-34$ ,  $-12.6$

Ordinary symbols : beta, BETA, Exp, Int

Special symbols : \$x, #9,13

Extended symbols :  $-3/2x$ ,  $212/113y$

Special system functions : \*, ^, :, Union, [ ]

Symbol naming convention : Upper and lower case characters are distinguished. System-defined symbols have their first character upper case and the rest lower case. User-defined symbols should avoid this form. Because of the special product input facility symbol names may not begin with digits. Symbols with arbitrary names may be formed by using ", ".

## Special Characters:

→ (i.e. return/line feed) : Input and process contents of present line, and print output from last segment.

; : Segment delimiter. Separates executable functions in expressions. If appears at end of line makes last command segment of line null, so no output is generated & from ↵ .

? ↵ : Provide assistance

! (as first character of line) : Designate remainder of line as a UNIX Shell command (e.g. invoking the editor).

\ (as last character of line) : Continue to next line

% : The most recently generated output line  
\$.

Special System Functions:

(Pass all arguments evaluated unless indicated otherwise.)

Arithmetic:

Infix (i.e. appear between their arguments, as in  $a * b$ ; prefix forms, such as  $\text{Mult}[a, b]$  given in parentheses) :

+ (Add)

\* (Mult)

/ (Div)

^ (Power)

. (Dot)

: inner product (of arbitrary conformable structures)

Postfix :

! (Fact)

!! (Dfact)

Note: — may be input as if it were an infix operator. In fact, it is always effectively unary (prefix) and is absorbed into rational coefficients.

\* need not be entered explicitly in such cases as  $2a$  ( $\rightarrow 2*a$ ) if no ambiguity exists.

Boolean:Infix:

- |        |           |                                                                                                                              |
|--------|-----------|------------------------------------------------------------------------------------------------------------------------------|
| >      | (Bgt)     | ( $a > b$ returns 1 if expression<br>a is known to be larger than b,<br>$\emptyset$ if is known not be, and DK<br>otherwise) |
| <      | (Blt)     |                                                                                                                              |
| $\geq$ | (Bge)     |                                                                                                                              |
| $\leq$ | (Ble)     |                                                                                                                              |
| $= =$  | (Beq)     | : Test for equality of values                                                                                                |
| $- =$  | (Bpeq)    | : Test for equality of properties                                                                                            |
| And    | (Band)    | : Boolean multiplication; if DK<br>appears in any term, DK returned                                                          |
| Or     | (Bior)    | : Inclusive Or                                                                                                               |
| For    | (Bear)    | : Exclusive or.                                                                                                              |
| In     | (Bsubset) | : A In B $\rightarrow$ 1 if A is<br>a subset of B and $\emptyset$ otherwise                                                  |
| Elem   | (Bellem)  | : A Elem B $\rightarrow$ 1 if A is<br>an element of B (a part at<br>level 1, and $\emptyset$ otherwise.                      |

Prefix :

~ (Not)

Not [arg]: If [arg] ==  $\phi$ , 1,  $\phi$

### List Manipulation:

Infix :

Union (Lunion) : The union of two lists  
(removing all occurrences of elements)

Inter (Linter) : The intersection (common elements) in two lists

To (Lto) : i To j gives the list of integers  $\{i, i+1, \dots, j-1, j\}$ .

Syntactic:Matchfix:

{ , }

(List)

(Internally treated as a null function)

Infix:

= (Eqn)

: Define equation (both sides evaluated), e.g. for use in Sub. Purely syntactic; no assignment actually performed

&amp; (Conc)

: General concatenator for suitably conformable structures

(E.g.  $a \& b \rightarrow ab$ ;  $\{a\} \& \{a, b\} \rightarrow \{a, a, b\}$ ,  
 $34 \& 12 \rightarrow 3412$ )Prefix:

' (Quote)

: Do not evaluate argument until Ev is explicitly applied at top level.

Fundamental :Infix :

: (Set)

(Simplify but do not scan l.h.s;  
evaluate and return r.h.s)

: Set value

:: (Qset)

(Simplify but do not scan l.h.s;  
pass r.h.s. unevaluated, but  
return evaluated r.h.s.)

: set value quoted

— (Prset) : set properties

Matchfix :[ , ] (Index) : Index into function, array,  
pattern, expression....

$A : B$ , where  $A$  and  $B$  are arbitrary expressions, will cause  $B$  to be substituted immediately for  $A$  when any expression containing  $A$  is referenced. For example, if

$a : b$

then  $a \wedge 2 - a$  becomes  $b \wedge 2 - b$ .

If now  $q \wedge 2 : qa$ , then  $q \wedge 2 - q$  becomes  $qa - q$ ; the substitution is performed whenever the literal form entered on the l.h.s. of : appears.

$A[b_1, b_2, \dots]$  specifies the part of the structure  $A$  indexed by the subscripts  $b_1, b_2, \dots$

In all cases,  $A[b_1][b_2][\dots]$  is equivalent to  $A[b_1, b_2, \dots]$ .

If  $A$  is an ordinary expression,  $A[b_1, b_2]$  may be interpreted as the  $b_2$ th part of the  $b_1$ th part of  $A$ . Hence, if  $A : a * x \wedge 2 + 213$ ,  
 $A[1] \rightarrow a * x \wedge 2$ ,  $A[2] \rightarrow 213$ ,  $A[1, 1] \rightarrow a$ ,  
 $A[1, 2, 2] \rightarrow 2$ ,  $A[2, 4] \rightarrow \text{None}$  etc.

If  $A$  as such has not been given a value, subscriptings of  $A$  may be interpreted as specifying elements of the array  $A$ , or the function  $A$  with particular arguments. Subscripted symbols may appear on the l.h.s. of set.

For example, to define the vector  $V : \{1, 0, 2\}$  one could instead perform

$$\begin{aligned} V[1] &: 1 \\ V[2] &: \emptyset \\ V[3] &: 2 \end{aligned}$$

$V$  would then have the value  $\{1, \emptyset, 2\}$ . However, the 'array'  $V$  may be sparse.  $V[2|2]$  could be given a value, as in

$$V[2|2] : x \wedge 2 + 3.$$

In fact, the subscripts may be arbitrary expressions, so that

$$V[x] : 15$$

$$V[x \wedge 2 - 2] : 18 + a.$$

After all these sets, the structure  $V$  would display as

$$\left. \begin{aligned} [1] &= 1, [2] = \emptyset, [3] = 2, [2|2] = x \wedge 2 + 3, \\ [x] &= 15, [x \wedge 2 - 2] = 18 + a \end{aligned} \right\}.$$

The assignments for  $V$  given above could be considered to provide simplification rules for the function  $V$ . If  $V$  has the specific literal argument  $x$ , then  $V[x] \rightarrow 15$ .

If the l.h.s. of a set is not a single symbol, but rather a subscripted symbol, then the symbol will henceforth be treated as an array or function name. Any expression may be considered as its leading operator subscripted by its arguments. For example,

$$a * b \rightarrow \text{Mult}[a, b]$$

and

$$a^{12} + 3/2 * c \rightarrow \text{Add}[a^{12}, 3/2 * c].$$

Thus an assignment such as  $a * b : c$  may be stored as  $\text{Mult}[a, b] : c$ .  $\text{Mult}$  would then display as  $\{[a, b] = c, \dots\}$

Note that if  $M : a^{13} + 3$ , then  $M[1, 2] : 5$  would cause  $M$  to become  $a^{15} + 3$ .

If all successive subscriptings of a symbol have been given values, then the symbol will be just a list (or vector) or a list of lists (or matrix / tensor), and will be displayed in the corresponding manner.

The subscripts discussed above have all been literal. To allow function definition, in the usual sense, dummy names beginning with \$ must be used. Hence, while  $f[x] : x^2$  does not define any substitution for  $f[y]$ , the assignment  $f[$x] : $x^2$  defines a function  $f$  such that for any argument  $$x$ ,  $f[$x]$  becomes  $$x^2$ . In this case,  $f[y]$  would become  $y^2$ . Dummy symbols allow the creating of user-defined functions. The subscripts of a symbol may be

a mixture of dummy and literal. Hence, for example  $P[1, \$x] : \$x$ ,  $P[2, \$x] : \frac{1}{2}(3\$x^2 - 1)$  defines an array of functions.  $P$  would display as

$$\{ [1, \$x] = \$x, [2, \$x] = \frac{1}{2}(3x^2 - 1) \}.$$

Functions may be defined recursively. Functions may return their original form; no infinite recursion will result (the `scan` function in the evaluator will tag the expression as not further evaluable).

The searching of the 'pattern trees' which form the values of subscripted objects is a costly procedure. All expressions will be simplified as much as possible by the evaluator before subscript searches begin.

The order of the search will be according to the first \* subscript, then the second. The most specific assignments for a given subscript will usually be placed first in the pattern. (This default behavior may be prevented by use of the third, optional, argument for Set).

Note that all evaluation is to infinite level. If  $a:b$  and  $b:c$ , then  $a$  becomes  $c$ , not  $b$ . Indirect assignments are not ratified, however, until the relevant symbols are referenced. For example, if  $a:b$ ,  $b:c$ ,  $b:d$  then

a would return d not c.

The property set function — allows a Boolean expression to be associated with a symbol. The expression is tested by use of - =.

The functions Pset and Pcl described below allow each of 10 binary properties to be set for a given symbol. (Another 6 binary properties are reserved for the use of system functions).

## Input / Output :

The prompt for the  $I_{12}$  input line will be

$I_{12} ::$

while the associated output line, if it is displayed, will begin with

$O_{12} : \dots$

These prefaces reveal that unevaluated input lines are assigned to the symbols  $I_i$ , while the evaluated output lines are given by  $O_i$ . To reexecute a given input line is achieved by  $EV[I_i]$ .

Whenever an input line  $I_i$  is processed, the CPU time taken is recorded in  $Time[i]$ .

## Basic System Functions (compulsory arguments underlined):

The special system function Set (or :) may have a third argument in its prefix form. Set [lhs, rhs, lev(:ϕ)] will perform lhs : rhs ensuring that in the pattern tree to which lhs belongs, all Set's with larger values of lev will be retrieved first.

Size [expr] : Number of storage units occupied by the complete tree of expr.

Dep(th) [expr] : Number of levels in the tree of expr.

Kill [expr<sub>1</sub><sup>\*</sup>, expr<sub>2</sub><sup>\*</sup>, ...] : Destroy values (and properties) assigned to the expr<sub>i</sub>.

Out [expr<sub>1</sub><sup>\*</sup>, expr<sub>2</sub><sup>\*</sup>, ...] : Place all details of the expr<sub>i</sub> in disk files, and leave only a single pointer in the working space.

App(ly) [func<sup>\*</sup>, list] : func [list[1], list[2], ...]

Evaluate [expr] : evaluate expr, ignoring all quotations.

\* Pass argument unevaluated.

If [pred, expr1, expr2, expr3] : If pred evaluates to 1<sup>t</sup>, return expr1, if to  $\emptyset$  expr2, and if to DK (for "Don't know") expr3.

For [begin, end, increment, expr] : Evaluate expr in a loop, starting with begin, ending with end.

Rpt (repeat) [expr, numb (:1)] : Evaluate expr numb times, returning the last value found.

Sub(stitute) [expr, list1, list2, ...] : Evaluate expr by successively attempting to apply the substitutions defined by the equations in list1, list2, ... .

Pset [symb, numb] : Set binary property number numb ( $\leq 10$ ) associated with symbol symb.

Pcl(ear) [symb, numb (:All)] : Clear binary property numb in symb.

Pt(est) [symb, numb] : Test binary property numb in symb.

+ In fact, to anything other than  $\emptyset$  or DK.

**Map** [f, expr, lev (:1)] : Structure obtained from expr by recursively replacing its subparts by f[subpart] in such a way that there are never more than lev nestings of the function f.

**Pos(ition)** [elem, expr] : List of subscripts corresponding to the first (in breadth-first search) occurrence of elem in expr (returns  $\emptyset$  if  $\sim \text{elem}$  In expr).

**Fill** [expr, elem (:φ)] : Fill the object expr with the element elem until expr consists of contiguous lists at all levels.

**Tree** [expr, lev (:Inf)] : A list of equations defining substitutions (for internally-generated symbols) necessary to construct expr from its level lev subparts. (A picture of the internal representation of expr.)

**Vars** [expr] : List of all symbols appearing in expr.

**Nur (numerical value)** [expr] : The numerical value of expr obtained by replacing rational by floating point numbers, and substituting numerical values for constants.

Hash [expr] : An integer hash code (with absolute value  $\leq 2^{31}$ ) for the expression expr.

Rand(om) [type (:Numb)] : A random object of specified type (default is floating point number  $0 \leq x \leq 1$ , but random expressions also available).

Delete) [expr, elem1, elem2, ...] : Remove all occurrences of the elemi from expr. (If necessary, remove the minimal subtrees which leave expr syntactically correct.)

Const(ants) [var1, var2, ...] : Declare vari to be constants, so that any value assigned to them will be assumed numerical, and accessed only by Nw.

## List Manipulation Functions:

`Sort[list, crit (:alphabetical order)]` : Place the elements of list in the order specified by applying the function crit to each of them.

`Cyc(left)[list, n(:1)]` : Cycle the elements of list  $\pm n$  positions to the right (left).

`Rev(erse)[list]` : Reverse the order of elements in list

## More obscure functions:

`Tup(les)[list, n(:1)]` : List of all possible ordered n-tuples of elements in list.

`Rearranging[list]` : List of all possible reorderings of list.

`Rep(licate)[list, n(:1)]` : List consisting of concatenation of list n times.

## Simplification Functions:

$\text{Dist}(\text{ribute}) [\underline{\text{expr}}, *_{\underline{\text{op2}}}, *_{\underline{\text{op1}}} (: \text{expr}[\text{Op}])]$  :

Distribute the function  $\text{op1}$  in  $\text{expr}$  over the function  $\text{op2}$ .

$\text{Tdist}(\text{ribute}) [\underline{\text{expr}}, *_{\underline{\text{op}}}, \text{pos} (: \{\text{Op}\})]$  : Distribute the function at position  $\text{pos}$  in  $\text{expr}$  over the function  $\text{op}$  at the next level down.

$\text{Ex}(\text{pand}) [\underline{\text{expr}}, \text{num} (: \text{Inf})]$  : Distribute the first  $\text{num}$  (in breadth-first scan) occurrences of  $*$  or  $/$  in  $\text{expr}$  over  $+$ .

$\text{Flat}(\text{ten}) [\underline{\text{expr}}, *_{\underline{\text{op}}}]$  : Flatten out all successive occurrences of the function  $\text{op}$  in  $\text{expr}$  (i.e. take  $\text{op}$  to be associative).

$\text{Tflat}(\text{ten}) [\underline{\text{expr}}, \text{pos} (: \{\text{Op}\})]$  : Flatten out the arguments of the function at position  $\text{pos}$  in  $\text{expr}$ .

$\text{Com}(\text{mute}) [\underline{\text{expr}}, *_{\underline{\text{op}}} (: \text{expr}[\text{Op}])]$  : Place the arguments of all occurrences of the function  $\text{op}$  in  $\text{expr}$  in alphabetical order (i.e. take  $\text{op}$  commutative).

$\text{Tcom}(\text{mute}) [\underline{\text{expr}}, \text{pos} (: \{\text{Op}\})]$  : Place the arguments of the function at position  $\text{pos}$  in  $\text{expr}$  in alphabetic order.

Remid (remove identity element) [expr, id(:1), \*op(:expr[Op])] :

Remove all literal occurrences of the element id from the argument lists of the function op in expr.

Reminv (remove inverses) [expr, \*inv, \*op(:expr[Op])] :

Remove all pairs of elements of the form  $\$x, \text{inv}[\$x]$  from the argument lists of the function op in expr.

Trans(cent) [expr, \*f, \*op(:expr[Op])] : Replace groups of  $\$n$  successive identical elements  $\$x$  in the argument lists of op in expr by  $f[\$x, \$n]$ .

Many of these functions are used directly in the internal simplifiers for system functions. Simplification functions for user-defined functions may be introduced by a high-level set.

## Algebraic Manipulation Functions:

**Len(gth)** [expr, lev (:ϕ)] : The maximum number of parts which could result from expansion of the expression expr with num = lev.

**Coef(ficient)** [expr, var, lev (:Inf)] : The sum of the coefficients of the subexpression var appearing in the form of expr obtained by Expand with num = lev.

**Hipow(er)** [expr, var, lev (:Inf)] : The highest power of var (or lowest of  $1/\text{var}$ ) appearing in the form of expr expanded with num = lev.

**Rat(ionalize)** [expr, list (:Vars [expr])] : Place all terms containing any of the symbols in list over a common denominator.

**Num(erator)** [expr] : The numerator of expr.

**Den(ominator)** [expr] : The denominator of expr.

**Conf(actor)** [expr] : Overall constant and numerical factors in expr.

**Fac(tor)** [expr, list (:Vars [expr])] : Simplify expr by extracting successively terms containing the symbols in list, and factoring them.

Comb(ine) [expr, list (: same denominators)] : Collect terms in expr which contain the various symbols in list.

Parf (partial fraction) [expr, var (: Vars[expr][1])] : Place expr in a partial fraction form with respect to the variable var.

Internal representation: (Numbers indicates bytes used)

Basic storage unit =  $4 \times 4$  bytes

Numbers:

hdr	flags		error estimate	double	$\longleftrightarrow$	float
1	1		4			8

Symbols:

hdr	flags	nargs	Table $\uparrow$	Coefficient Numerator	Coefficient Denominator	Arguments (if any) $\rightarrow$
1	1	2	4	4	4	

Limbs:

hdr	flags	LEPC	Tree $\uparrow$	Index $\uparrow$	Hash of highest subtree
1	1	2			

Table entry:

hdr	flags	Binary properties	Property Extension $\uparrow$	LEPC control	Print name $\uparrow$	Value / Pattern tree $\rightarrow$
1	1	2	4	4	4	

In table entries for system-defined functions, insert  
Table entry extension:

hdr	flags		Text routine $\uparrow$		
1	1		4		

Chain link:

hdr	flags	Access counter	This Link $\uparrow$	Next Link $\uparrow$	Previous Link $\uparrow$
1	1	2	4	4	4

In each case, the header `hdr` defines the type of the storage unit, and contains information for the storage allocator.

$\uparrow$  denotes a pointer.

Symbols are found by hashing into a contiguous table starter space (of length 128 units) which contains the first links in chains of table entries.

The table entries contain pointers to the symbol print names, which are stored in a contiguous heap of memory.

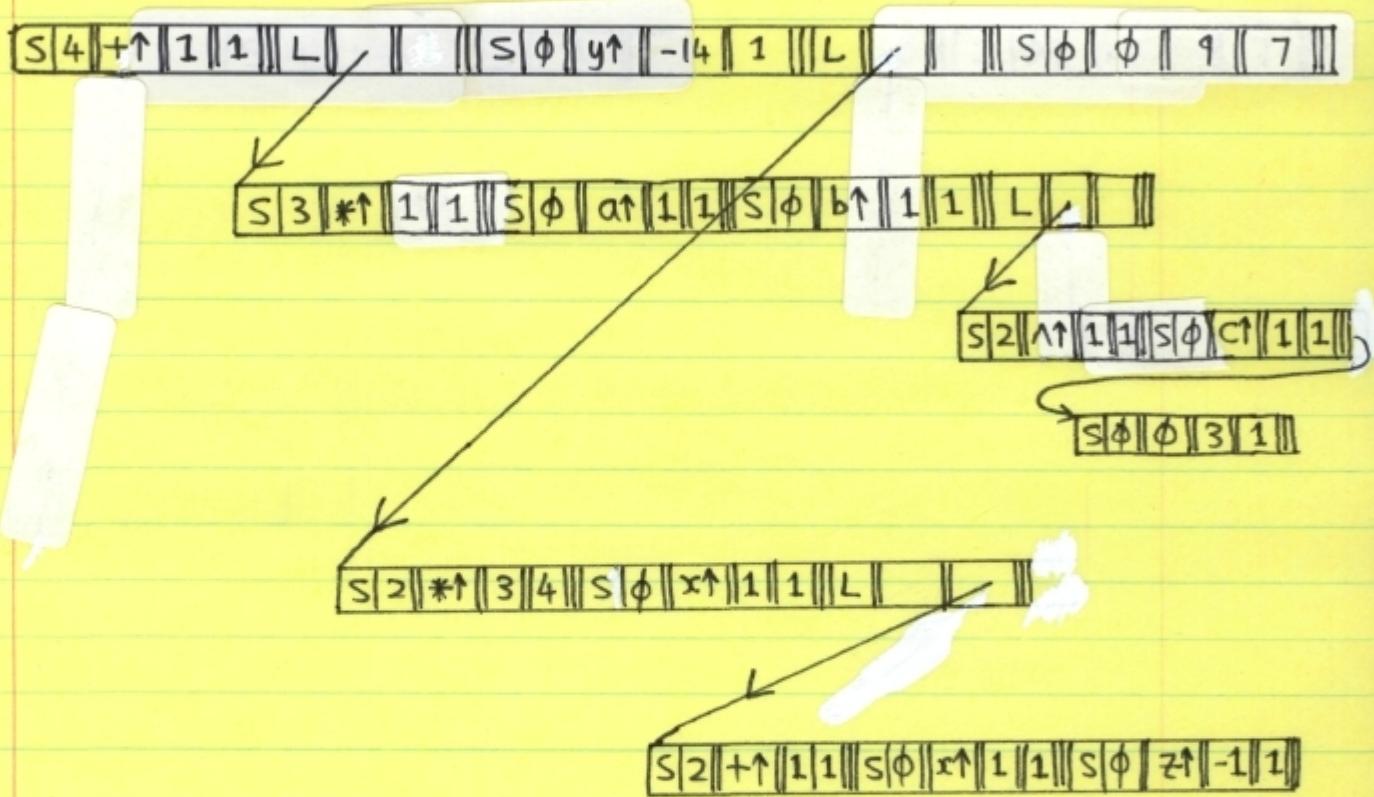
The LEPC (Last Evaluator Pass Counter) prevents unnecessary reevaluation of subexpressions.

The text routine  $\uparrow$  points to the entry of a C object module.

Whenever an expression is used, it is copied. Only table entries are multiply referenced. The storage allocator is immediately notified of the positions of freed storage units.

Example :

$$a * b * c \wedge 3 - 14y + \frac{3}{4}x(x - z) + \frac{9}{7}$$



The pointers are to the corresponding table entries.

Lists (and hence pattern trees, etc) are stored just like functions, except that they have no symbol header: if the subscripts required are not sequential, they are pointed to by the index↑ in the limbs.

## Mathematical functions

Abs [ $\underline{x}$ ] : Absolute value of  $\underline{x}$  (or modulus if  $\underline{x}$  is complex)

Trunc [ $\underline{x}$ ] : Integer part of  $\underline{x}$

Sign [ $\underline{x}$ ] : +1 for  $\underline{x} > 0$ , -1 for  $\underline{x} < 0$ , 0 for  $\underline{x} = 0$

Max [ $\underline{x}_1, \underline{x}_2, \dots$ ] : The maximum of the  $\underline{x}_i$

Min [ $\underline{x}_1, \underline{x}_2, \dots$ ] : The minimum of the  $\underline{x}_i$

Re(al) [ $\underline{x}$ ] : The real part of  $\underline{x}$

Im(aginary) [ $\underline{x}$ ] : The imaginary part of  $\underline{x}$

Arg [ $\underline{x}$ ] : The principal phase of  $\underline{x}$  ( $: \underline{\text{Atan}}[\text{Re}[\underline{x}]/\text{Im}[\underline{x}]]$ )

Exp [ $\underline{x}$ ] : Exponential of  $\underline{x}$  ( $: \underline{E} \wedge \underline{x}$ )

Log [ $\underline{x}$ , base ( $: E$ )] : Principal logarithm of  $\underline{x}$  to given base

Sqrt [ $\underline{x}$ ] : Principal square root of  $\underline{x}$  (with  $\underline{\text{Re}}[\text{Sqrt}[x]] \geq \phi$ )

Gamma [ $\underline{x}, a(\phi)$ ] : The (in general incomplete) Euler  $\Gamma$  function ( $\Gamma(x, a) = \int_a^\infty e^{-t} t^{x-1} dt$ )

Beta [ $\underline{x}, y, a(\phi)$ ] : The (in general incomplete) Euler Beta function

Psi [ $\underline{n}, \underline{x}$ ] : The polygamma function ( $\psi^{(n)}(x) = \frac{d^{n+1}}{dx^{n+1}} \log(\Gamma(x))$ )

Zeta [ $\underline{x}, a(1)$ ] : The (in general generalized) Riemann Zeta function ( $\zeta(x, a) = \sum_{n=0}^{\infty} \frac{1}{(n+a)^x}$ )

Li [ $\underline{n}, \underline{x}$ ] : The polylogarithm function ( $L_n(x) = \sum_{i=0}^{\infty} \frac{x^i}{i^n}$ )

Nielsen [ $\underline{n}, p, \underline{x}$ ] : The generalized Nielsen function

Comb [ $\underline{x}, y$ ] : The binomial coefficient  $\binom{x}{y}$

Pochhammer [ $\underline{x}, \underline{n}$ ] : The Pochhammer symbol  $\Gamma(x+n)/\Gamma(x)$

Ti [ $\underline{n}, \underline{x}$ ] : The imaginary polylogarithm function

<u><math>\text{Sin}[x]</math></u>	<u><math>\text{Asin}[x]</math></u>	<u><math>\text{Sinh}[x]</math></u>	<u><math>\text{Asinh}[x]</math></u>
<u><math>\text{Cos}[x]</math></u>	<u><math>\text{Acos}[x]</math></u>	<u><math>\text{Cosh}[x]</math></u>	<u><math>\text{Acosh}[x]</math></u>
<u><math>\text{Tan}[x]</math></u>	<u><math>\text{Atan}[x]</math></u>	<u><math>\text{Tanh}[x]</math></u>	<u><math>\text{Atanh}[x]</math></u>
<u><math>\text{Cot}[x]</math></u>	<u><math>\text{Acot}[x]</math></u>	<u><math>\text{Coth}[x]</math></u>	<u><math>\text{Acoth}[x]</math></u>
<u><math>\text{Sec}[x]</math></u>	<u><math>\text{Asec}[x]</math></u>	<u><math>\text{Sech}[x]</math></u>	<u><math>\text{Asech}[x]</math></u>
<u><math>\text{Csc}[x]</math></u>	<u><math>\text{Acsc}[x]</math></u>	<u><math>\text{Csch}[x]</math></u>	<u><math>\text{Acsch}[x]</math></u>

$\text{Gd}[x]$  : The gudermannian  $\text{Gd}[x] \equiv (\text{atan}[\text{Exp}[x]] - \pi i / 2)$   
 (Relates circular & hyperbolic functions, e.g.  
 $\sinh(x) = \tan(\text{gd}(x))$ ).

$\text{Bern}[n, x(:\phi)]$  : The  $n^{\text{th}}$  Bernoulli polynomial  $B_n(x)$   
 (Default is  $n^{\text{th}}$  Bernoulli number  $B_n \equiv B_n(0)$ )

$\text{Euler}[n, x(:1/2)]$  : The  $n^{\text{th}}$  Euler polynomial  $E_n(x)$   
 (Default is  $n^{\text{th}}$  Euler number  $E_n \equiv E_n(1/2)$ )\*

---

\* Sometimes,  $E_n \rightarrow 2^n E_n$ , which is just silly.

$Ei[z]$  : The  $n$ th exponential integral  
 $(Ei_n(x) = \int_1^{\infty} \frac{e^{-zt}}{t^n} dt)$

$Si[z]$  : The sine integral  $(Si(x) = \int_0^x \frac{\sin t}{t} dt)$

$Ci[z]$  : The cosine integral

$Logi[z]$  : The logarithm integral  $(li(x) = Ei(\log(x)) = \int_0^x \frac{dt}{\log t})$

$Erf[z]$  : The error function  $(erf(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt)$

$Erfc[z]$  : The complementary error function  $(Erfc(x) = 1 - Erf(x))$

$Frec[z]$  : The Fresnel integral  $C(x) = \int_0^x \cos\left(\frac{\pi}{2}t^2\right) dt$

$Fres[z]$  : The Fresnel integral  $S(x) = \int_0^x \sin\left(\frac{\pi}{2}t^2\right) dt$

$\text{Legp} [n, \pm, m(\phi)]$  : The (in general associated) Legendre polynomial  $P_n^m(x)$

$\text{Legq} [n, \pm, m(\phi)]$  : The (in general associated) second-kind (irregular) Legendre polynomial  $Q_n^m(x)$

$\text{Besj} [n, \pm]$  : The  $n^{\text{th}}$  cylindrical ordinary Bessel function  $J_n(x)$

$\text{Besy} [n, \pm]$  : The irregular Bessel function  $Y_n(x)$

$\text{Besh1} [n, \pm]$  : The complex Bessel function  $H_n^{(1)}(x)$

$\text{Besh2} [n, \pm]$  : The complex Bessel function  $H_n^{(2)}(x)$

$\text{Besk} [n, \pm]$  : The modified Bessel function  $K_n(x)$

$\text{Besi} [n, \pm]$  : The irregular modified Bessel function  $I_n(x)$

$\text{Besn} [n, \pm]$  : The Neumann function  $N_n(x)$

$\text{Besjs} [n, \pm]$  : The spherical Bessel function  $j_n(x)$

$\text{Besys} [n, \pm]$  : The spherical Bessel function  $y_n(x)$

$\text{Spha} [l, m, \pm, \phi(\phi)]$  : The spherical harmonic  $Y_l^m(x, \phi)$

$\text{Wigner} [\{l_1, l_2, l_3\}, \{m_1, m_2, m_3\}]$  : The Wigner 3-j symbol  $(l_1 l_2 l_3; m_1 m_2 m_3)$

$\text{Struve}[\underline{n}, \underline{x}]$  : The Struve function  $H_n(x)$

$\text{Ai}[\underline{n}, \underline{x}]$  : The Airy function  $Ai_n(x)$  (Third-integer order Bessel function)

$\left. \begin{matrix} S_n[\underline{x}, \underline{m}] \\ C_n[\underline{x}, \underline{m}] \\ D_n[\underline{x}, \underline{m}] \end{matrix} \right\}$  : The Jacobian elliptic functions

$\text{EllR}[\underline{k}, t(: \pi/2)]$  : The first-kind elliptic integral  
( $K(k) = F(\frac{\pi}{2}, k)$ )

$\text{EllE}[\underline{k}, t(: \pi/2)]$  : The second-kind elliptic integral

$\left. \begin{matrix} Ce[\underline{n}, \underline{x}, \underline{q}] \\ Se[\underline{n}, \underline{x}, \underline{q}] \end{matrix} \right\}$  : The Mathieu functions

$\text{Geg}[\underline{n}, \underline{x}, \underline{l} (:1)]$  : The Gegenbauer polynomials  $C_n^{\lambda}(x)$   
(Default is Chebyshev polynomials)

$\text{Herm}[\underline{n}, \underline{x}]$  : The Hermite polynomials  $H_n(x)$

$\text{Lag}[\underline{n}, \underline{x}, \underline{\alpha} (: \phi)]$  : The Laguerre polynomials  $L_n^{\alpha}(x)$

$\text{Jac}[\underline{n}, \underline{x}, \underline{\alpha}, \underline{b}]$  : The Jacobi polynomials  $P_n^{(\alpha, \beta)}(x)$   
(could be amalgamated with Legendre polynomials)

Hg11 [ $a, b, x$ ] : The Confluent Hypergeometric function  ${}_1F_1(a, b; x)$

Hg21 [ $a, b, c, x$ ] : The Gauss Hypergeometric function  ${}_2F_1(a, b; c; x)$

Coulf [ $\ell, e, r$ ] : The regular Coulomb wave function  $F_\ell(\eta, \rho)$ .

Coulg [ $\ell, e, r$ ] : The irregular Coulomb wave function  $G_\ell(\eta, \rho)$ .

Prime [ $\underline{n}$ ] : The  $n^{\text{th}}$  prime number

Stirling [ $\underline{n}$ ] : The  $n^{\text{th}}$  Stirling number

Gwgt [ $i, n(:6)$ ] : { $i^{\text{th}}$  sample point,  $i^{\text{th}}$  weight}  
for  $n$ -point Gaussian quadrature

$\Delta[\{i_1, i_2, \dots\}, \{j_1, j_2, \dots\}]$  : The Kronecker delta  
 $\delta_{i_1 i_2 \dots}^{j_1 j_2 \dots}$

$\epsilon[i_1, i_2, \dots]$  : The Levi-Civita symbol  $\epsilon_{i_1 i_2 \dots}$

$G[p_1, p_2, \dots]$  : The product of Dirac gamma matrices  
 $p^1 p^2 \dots$

$I_d[n, m(:n)]$  : The  $n \times m$  identity matrix

## Mathematical operations

Diff [ $f$ ,  $\underline{x_1}, x_2 \dots$ ] : The partial derivative of  $f$  with respect to  $x_1, x_2 \dots$

( $\text{Diff}[f, \{x, 3\}] = \frac{\partial^3 f}{\partial x^3}$ ; dependencies ignored unless displayed explicitly)

Difft [ $f$ ,  $\underline{x_1}, x_2, \dots$ ] : The total derivative of  $f$  with respect to  $x_1, x_2, \dots$

Int [ $f$ ,  $\underline{x}, x_1, x_2 \dots$ ] : The integral of  $f$ , indefinitely, or between limits  $x=x_1, x=x_2$ , possibly multidimensional.

Tay [ $f$ ,  $\underline{x}, pt, ord(\phi)$ ] : The Taylor expansion of  $f$  with respect to  $x$  about  $x=pt$  up to terms of order  $x^{ord}$ .

Lim [ $f$ ,  $\underline{x}, pt$ ] :  $\text{Tay}[f, \underline{x}, pt, \phi]$  (Limit of  $f$  as  $x \rightarrow pt$ ).

Sol(ve) [ $\{x_1, x_2, \dots\}, \{eqn_1, eqn_2, \dots\}$ ] : Solve for  $x_1, x_2, \dots$  in the equations ( $=$ 's)  $eqn_1, eqn_2, \dots$

Sum [ $f, i, i1(\phi), i2(:inf)$ ] :  $\sum_{i=i1}^{i=i2} f(i)$

Prod [ $f, i, i1(\phi), i2(:inf)$ ] :  $\prod_{i=i1}^{i=i2} f(i)$

Nint [f, x, x1, x2] : Numerical integration  $\int_{x_1}^{x_2} f dx$

Nsol [{x1, x2, ...}, {eqn1, eqn2, ...}] : Numerical solution

Nsum [f, i, i1(:phi), i2(:Inf)] : Numerical summation

Nprod [f, i, i1(:phi), i2(:Inf)] : Numerical product

Det [mat, {i1, i2, ...}] : The determinant of matrix mat, or its i1, i2, ... th cofactor

Minor [mat, {i1, i2, ...}] : The i1, i2, ... th minor of mat.

Eival [mat] : The eigenvalues of mat

Eivec [mat] : The eigenvectors of mat

Simtran [mat] : The matrix U such that  $U^{-1} \cdot \text{mat} \cdot U$  is diagonal (matrix of eigenvectors).

Div [f, {x1, x2, ...}] : The divergence of the vector function (list) f with respect to the Cartesian coords x1, x2, ...

Grad [f, {x1, x2, ...}] : The gradient of f

Curl [f, {x1, x2, x3}] : The curl of f

Lap [f, {x1, x2, ...}] : The Laplacian ( $\square$ 'Alemberian, ...) of f.

## Simplification rules

Elementary simplifications, such as  $a*b*1 \rightarrow a*b$  or  $\text{Exp}[a]*\text{Exp}[b] \rightarrow \text{Exp}[a+b]$  will be applied automatically by the evaluator. More complicated simplifications, such as  $\text{Zeta}[2] \rightarrow \pi^2/6$  or.  $G[\mu, a, b, c, \mu] \rightarrow -2*G[c, b, a]$ , must be applied selectively. A human and machine readable book of simplifications of all kinds will be generated. Each simplification will be given a name, and will be of the form e.g.

$$\underline{\text{SSi1 : } \sin[2*\$x] = 2*\sin[\$x]*\cos[\$x]}$$

The rule can be applied to an expression exp simply by doing Sub[exp, SSi1]. Sets of simplification rules, such as those for expanding trigonometric functions with compound arguments, will be given collective names, so that, for example, Trigex: {SSi1, SSi2, ... (other trigonometric expansion rules)}. Application of all these rules is achieved by Sub[exp, Trigex]. The book of simplification rules should contain essentially all known identities pertaining to special functions, tensor and spinor algebra, etc. Eventually, some heuristic programs which decide which rules to apply to a given expression may be devised. The more controlled approach in which the user locates the rule in the simplification book, and applies it explicitly, seems, nevertheless, to be preferable, in most circumstances.

$\text{Tr}[\underline{f}]$  : The Dirac trace of  $f$

$\text{Con}[\underline{f}]$  : Contract paired indices in  $f$