Chapter 5: Monadic Scalar Functions

As stated in the introduction, the term *integer* is used in this manual to indicate not only a domain of values but also a particular internal representation. To refer to the same domain of values when both integer and floating point representations are allowed, the term *restricted whole number* is used. These floating point representations need only be tolerably equal to the integers.

Classification of Monadic Scalar Functions

Although they are listed alphabetically in this chapter, for convenient reference, the A+ monadic scalar primitive functions can be grouped—among other ways, to be sure—in four categories:

- the most common arithmetical functions: Reciprocal, Negate, Identity;
- other arithmetical functions: Exponential, Natural log, Pi times, Roll;
- extractive functions: Sign, Absolute value, Floor, Ceiling;
- logical function: Not.

Application and Result Shape

All monadic scalar functions produce scalars from scalars, and apply element by element to their arguments: they are applied to each element independently of the others. Consequently, the shape of the result is the same as the shape of the argument. This behavior is assumed in the following descriptions.

Error Reports

Multiple errors elicit but a single report. With only one exception, the error reports for monadic primitive scalar functions are common to all such functions. There are six reports, including interrupt, and each error report on the following list is issued only if none of the preceding ones apply:

- parse: this error class includes valence errors, which must result from three or more arguments in braces, since every symbol for a monadic scalar primitive function is also used for a dyadic function;
- value: the argument has no value;
- nondata: the argument is a function or some other nondata object;
- type: the argument is not a simple numeric array—for Not, of restricted whole numbers, and, for Natural Log, of nonnegative numbers—; the Identity function, however, cannot cause this error report;
- wsfull: the workspace is currently not large enough for execution of the function; a bare left arrow (ß), which dictates resumption of execution, causes the workspace to be enlarged if possible;
- interrupt (not an error): the user pressed **c** twice (once if A+ was started from a shell) while holding the **Control** key down.

An inadvertent left argument results not in a valence error, but in the invocation of a dyadic function that shares the function symbol.

Absolute value ´ x

Argument and Result

The argument and result are simple numeric arrays. In Version 2, the result is always floating point. In Version 4, the result for an integer argument is integer if possible.

Definition

The absolute value of x. In other words, ´ x is equivalent to x times Sign of x.

Example

```
     ´12.3 ¢3
 12.3 3
```

Ceiling x

Argument and Result

The argument and result are simple numeric arrays. The result consists of nonfractional numbers, and is integer if all its elements can be represented that way (including if empty). If some element of the result has too great a magnitude to be represented as an integer, the result is floating point.

Dependency

Comparison tolerance, for most floating point numbers (see “Comparison Tolerance”, page 105).

Definition

The smallest nonfractional number greater than x or tolerably equal to x, except that x is 0 when x exceeds zero but is equal to or less than `1e-13` (intolerantly).

Example

```
      10 10.2 10.5 10.98 ¢9 ¢9.2 ¢9.5 ¢9.98, 10+1e-13
 10 11 11 11 ¢9 ¢9 ¢9 ¢9 10
```

Exponential *x

Argument and Result

The argument and result are simple numeric arrays. The result is always floating point.

Definition

e (2.71828...) to the power x.

Example

```
     *¢1 0 1 2 710
 .3678794412 1 2.718281828 7.389056099 Inf
```

Floor ~x

Argument and Result

The argument and result are simple numeric arrays. The result consists of nonfractional numbers, and is integer if all its elements can be represented that way (including if empty), else floating point.

Dependency

Comparison tolerance, for most floating point numbers (see “Comparison Tolerance”, page 105).

Definition

The largest nonfractional number less than x or tolerably equal to x, except that ~x is 0 when x is less than zero but is equal to or greater than ¢1e-13 (intolerantly).

Example

```
     ~10 10.2 10.5 10.98 ¢9 ¢9.2 ¢9.5 ¢9.98, 10-1e-13
10 10 10 10 ¢9 ¢10 ¢10 ¢10 10
```

Identity +x

Argument and Result

The argument, which is also the result, can be any array. (A type error cannot occur.)

Definition

The result is identical to x.

Example

```
     +'abc'
abc
```

Natural log x

Argument and Result

The argument and result are simple numeric arrays. The elements of the argument must be nonnegative. The result is always floating point.

Definition

The natural logarithm of x, i.e., the logarithm of x to the base *e* (2.71828...).

Example

```
       1 10 100 0
0 2.302585093 4.605170186 ¢Inf
```

Negate -x

Argument and Result

The argument and result are simple numeric arrays. In Version 2, the result is always floating point. In Version 4, the result for an integer argument is integer if possible.

Definition

`0-x`.

Example

```
     -23 ¢2 45 0 ¢1 .5
 ¢23 2 ¢45 0 1 ¢0.5
```

Not ~x

Argument and Result

The argument is a simple array of restricted whole numbers. The result is always of integer type.

Definition

The value is 1 if ~ x is almost 0, viz., less than `1e-13` (intolerantly), and 0 otherwise.

Examples

```
     ~0 1
 1 0
     ~¢1 0 1 2 3
 0 1 0 0 0
```

Pi times ˇx

Argument and Result

The argument and result are simple numeric arrays. The result is always floating point.

Definition

Pi (3.14159...) times `x`. The result is `Inf` or `¢Inf` if it cannot be represented otherwise.

Example

```
     ˇ1 2 .5 1e308
 3.141592654 6.283185307 1.570796327 Inf
```

Reciprocal x

Argument and Result

The argument and result are simple numeric arrays. The type of the result is always floating point.

Definition

`1 x`. The result is `Inf` or `¢Inf` for elements that cannot be represented otherwise; in particular, the result is `Inf` for 0.

Example

```
      .5 1.5 ¢2 100 0 ¢1e-309
 2 0.6666666667 ¢0.5 0.01 Inf ¢Inf
```

Roll ?x

Argument and Result

The argument and result are simple arrays of restricted whole numbers. The result is always integer.

Dependency

The value of the Random Link system variable, `'rl` (page 132), which is changed each time a random number is chosen.

Definition

`x` is an array of positive restricted whole numbers, and the value is an array of integers with the same shape as `x`. Each element of the result is a random integer chosen from `e`, where `e` is the corresponding element of `x`. The result is dependent on the random link, `'rl`, which is set when the Random Link system command, `$rl`, (page 174) is executed and each time a random integer is chosen.

Example

```
     ?20 10
 3 4 7 2 1 5 0 1 7 0 7 9 9 6 9 3 9 0 0 6
     ?20 10
 2 9 2 4 5 1 3 5 8 3 0 3 7 8 6 9 5 8 0 4
```

Sign «x

Argument and Result

The argument and result are simple numeric arrays. The type of the result is always integer.

Definition

Signum `x`. The value of `«x` is -1 for negative elements, 0 for zero, and 1 for positive elements.

Example

```
     «100 ¢2.5 0 5 ¢Inf
 1 ¢1 0 1 ¢1
```