Laconic Operation Documentation

Adam Yedidia

April 11, 2016

This document enumerates all the primitive operations in Laconic. It is recommended that users start by reading laconic_quick_start.pdf to get a sense for how Laconic works.

Remember that operations can be combined into complex expressions (i.e. (a+b)*c) but that full parenthesization is required! Also, recall that ints are signed integers with no maximum or minimum value, lists are Python-style lists of ints, and list2s are Python-style lists of lists.

1 Integer Operations

Assume in the operations below that x and y are variables of type int with values x and y, respectively.

All the operations below yield values of type int.

1.1 Addition

The expression x+y yields the value $x + y$.

1.2 Subtraction

The expression x-y yields the value $x - y$.

1.3 Multiplication

The expression x*y yields the value xy.

1.4 Integer Division

The expression `x/y` yields the value $s(xy)\left\lfloor \left|\frac{x}{y}\right| \right\rfloor$, where $s()$ is the sign function. In plain English, integer division in Laconic rounds numbers to the lowest-magnitude adjacent number. This means that `3/2` would yield the value 1, and `(0-3)/2` would yield the value -1.

If `y` is 0, Laconic will throw a runtime error if interpreted, and the compiled TMD or Turing machine will enter an infinite loop.

1.5 Negation

The expression ~x yields the value $-x$. Note the strange negation operator.

1.6 Equality

The expression `x==y` yields the value 1 if $x = y$, and the value 0 otherwise.

1.7 Inequality

The expression `x!=y` yields the value 1 if $x \neq y$, and the value 0 otherwise.

1.8 Greater Than

The expression `x>y` yields the value 1 if $x > y$, and the value 0 otherwise.

1.9 Less Than

The expression `x<y` yields the value 1 if $x < y$, and the value 0 otherwise.

1.10 Greater or Equal

The expression `x>=y` yields the value 1 if $x \geq y$, and the value 0 otherwise.

1.11 Less Than or Equal

The expression `x<=y` yields the value 1 if $x \leq y$, and the value 0 otherwise.

1.12 And

The expression `x&y` yields the value 1 if $x > 0$ and $y > 0$, and the value 0 otherwise. Note that negative values of x and y are interpreted as “false” values for the purposes of “boolean” operations.

1.13 Or

The expression `x|y` yields the value 1 if $x > 0$ or $y > 0$, and the value 0 otherwise.

1.14 Not

The expression `!x` yields the value 1 if $x \leq 0$, and the value 0 otherwise.

2 List and List2 Operations

Assume in the operations below that `x` is a variable of type `int`, `l` is a variable of type `list`, and that `L` is a variable of type `list2`. Assume that these variables have values of x, l, and L, respectively.

2.1 Indexing

The expression `l@x` yields the `int` value of the x^{th} element of l, assuming 0-indexing. If $x \leq |l|$, a runtime error is thrown in both the interpreted and compiled versions of the code.

The expression `L@*x` yields the `list` value of the x^{th} element of L, assuming 0-indexing. If $x \leq |L|$, a runtime error is thrown in both the interpreted and compiled versions of the code.

In general, `list2` operations use the same symbol as the corresponding `list` operations, but with a `*` at the end.

2.2 Appending

The expression `l^x` yields the `list` value of $l||[x]$, where $||$ denotes the concatenation operation.

The expression `L^*l` yields the `list2` value of $L||[l]$, where $||$ denotes the concatenation operation.

2.3 Length

The expression `#l` yields the `int` value of $|l|$.

The expression `#*L` yields the `int` value of $|L|$.

2.4 Concatenation

In the expressions below, the variables `l1`, `l2`, `L1`, and `L2` have types `list`, `list`, `list2`, and `list2`, with values of l_1, l_2, L_1, and L_2, respectively.

The expression `l1||l2` yields the `list` value of $l_1||l_2$, where $||$ denotes the concatenation operation.

The expression `L1||*L2` yields the `list2` value of $L_1||L_2$, where $||$ denotes the concatenation operation.