Laconic Quick Start Documentation

Adam Yedidia

April 10, 2016

Laconic is an imperative language. Users accomplish what they wish by declaring and modifying variables using sequences of commands that are separated by semicolons. Users may also define functions, which behave similarly to how they do in most languages.

1 Data Types

There are three types of variables: `int`s, `list`s, and `list2`s. An `int` is a signed integer, a `list` is a list of `int`s (like in Python), and a `list2` is a list of `list`s.

To declare a variable, write the data type followed by the variable name, for example:

```
int myInteger;
```

2 Variable Assignment and Manipulation

Variables are assigned to values like in most languages, with the = operator. Variables can be combined in complex expressions, i.e. `(a+b)*c`, but full parenthesization is required.

Variable assignment to `list`s or `list2`s automatically copies the relevant value, instead of assigning to a pointer like in Python.

3 Functions

Functions are defined with the keyword `func`. Like in Java, the function arguments are contained in parentheses, and the function body is contained in curly-braces (unlike in Java, argument type is not specified). For example:

```
func addY(x, y) {
    x = x + y;
    return;
}
```

Functions are called with statements such as (for example) "`addY(a, b);`".

Functions are different from how they are in most programming languages in a few important ways.

1. Functions have no return value. They effect change by changing the values of the variables that are passed into the function. All variables passed into the function, including `int`s, are subject to modification by the function.

2. Only variables can be passed into functions, and each variable can only be passed in once. (So commands like "`addY(a, 5);`" or "`addY(a, a);`" are illegal.)

3. Functions do not have internal variables (so variables cannot be defined inside functions). Any variables needed to perform computation must be passed in as arguments, even if they are only useful for temporary storage.

4 If and While Statements

Laconic supports `if` and `while` statements. Syntax is the same as in Java or C, (for example):

```
while (x < y) {x = x+1;}
```

5 Halting

To halt execution (and make the Turing machine that the code compiles to halt), use the `halt` command. For example, "`halt;`". Halt commands may not appear within functions.

6 Using the Program

See `laconic_readme.pdf` for an explanation for how to compile or interpret a Laconic program. The directory at:

`parsimony/src/laconic/laconic_files/`

contains examples of Laconic programs.