Course Project 3 - Esoteric Language Design

Devore-McDonald, Breãnna
breanna-devore-mcdonald.herokuapp.com

Jones, Nicholas
nicholascjones.com

April 20, 2016

Abstract

For this project, we created an esoteric, Turing-complete language called *xpl3t!v3* (*read: "expletive"*). Our code can be found at https://github.com/bdevorem/expletive , as well as at our website, http://expletive.herokuapp.com/ .

1 Relevance

This project is relevant to the class because we designed and wrote an interpreter for a language to represent a Turing machine using a model from early computational theory.

2 Effort

The two of us worked in a pair on this project. Our work consisted of performing basic research on programmatic implementation of Turing machines, designing the structure of the language, writing an interpreter for the language in Python, writing example programs for the language, and creating a website for documentation and release to the public. Breanna spent 8 hours on the project, and Nicholas spent 7 hours.

3 Content

3.1 Theory

When deciding on the direction of our project, we spent much of our time attempting to decide on a computational model. We eventually decided on a deterministic model known as the 2-tag system, first proposed in 1943 by Emil Leon Post. This model is Turing-complete, as proved by the fact that it can simulate the Universal Turing Machine (*Wang 74*). It makes use of a FIFO (first-in, first-out) queue as the machine's tape (*"Tag system" 2016*).

As expressed in Post's terms, this model relies on relationships between "primitive letters at one end" and those at the other end of a sequence, known as 'productions' (*Post 202*). In less formal, more modern terms, the mechanism

consists of a relatively simple process. At each computation, the first two elements of the queue are popped from the queue. The productions, referred to as "rules," consist of pushing an arbitrary number of symbols to the end of the queue, based on the first popped symbol. For example, in the example string "abc", if there is a rule relating the popped symbol "a" to pushed sequence "de", this computation will end with the queue as "cde". Computation will halt when coming upon an arbitrary "halting symbol", or when the queue is at size 1 or less. The fact that a two-tag system (or any system with more than one tag can simulate any Turing machine comes from the fact that the queue can significantly grow or shrink at any point, by either a reduction of two or one, no change in length, or an addition of varying length, based on the rules defined (Weisstein 2016).

3.2 Program

For this project, we used Python to write an interpreter, referred to as xpl.py. The code is stored in a GitHub repository (https://github.com/bdevorem/expletive). All directions for execution are included in the README.md folder.

The interpreter, xpl.py, which is written in Python, follows a basic structure to express expletive computation through Python execution. First, the program takes two command line arguments, the -i flag (for initializing the initial string) and the -v flag (for a verbose mode), which slightly modify program execution. The interpreter then gets all of the lines of the program, and stores them for execution. Each line's first word is then read for commands using a series of if statements. If the command is the "@|?h" command ("alph"), the included characters are stored as the tape alphabet. The "h@|+" command ("halt") is used to give the halting character on the tape. The "r#|3" command ("rule") stores the first character given in as the key to a dictionary entry, with the value being the second set of characters given. The "n?#t" command ("input") is used to initialize the input string programatically, rather than on the command line. Lastly, the "+|+1e" command ("title") is used to give a printed title to the program. After the rules are established, which make up the bulk of the program, the execution runs a while loop that pushes and pops values from the tape as defined in the rules. If verbose mode is included, the queue is displayed at each step of computation; if it is not used, the end queue is all that is printed after the title.

Also included in the repository of note are scripts written in expletive (in the scripts folder). Scripts include a test.xpl, an implementation of Wikipedia's example of a tag system, hello.xpl, and a "Hello, World" program. Additionally, we wrote a program, infinite.xpl, that loops forever by appending and popping the same amount of the same character from the queue. Lastly, we wrote a program called coins.xpl that takes an input of types of coins and outputs a number of x's equal to the number of cents calculated.

4 Bibliography

"Tag system." Wikipedia: The Free Encyclopedia. Wikimedia Foundation, Inc. 9 March 2016. Web. 19 April 2016. → Used for reference on how the tag system works.

Post, Emil L.. "Formal Reductions of the General Combinatorial Decision Problem". American Journal of Mathematics 65.2 (1943): 197–215. Web 19 April 2016 → Paper in which tag system is expressed, used for formal definition and description of system.

Wang, Hao. "Tag Systems and Lag Systems." Mathematische Annalen Math. Ann. 152.1 (1963): 65-74. Web. 19 Apr. 2016. → Used to show that 2-Tag system is Turing-complete.

Weisstein, Eric W. "Tag System." From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/TagSystem.html → Used to understand theory behind tag systems