Biography - Independent Study Ideas

DeDominic, Anthony
Eastern Connecticut State University
Willimantic, USA
dedominica@my.easternct.edu

Abstract

This paper will introduce me, the author, by including biographical information. Information such as, where I was born, about my brother. It will also discuss why I chose to study at ECSU and what interested me about computer science. I will also propose a few ideas that I desire to research for my undergraduate capstone; Ideas like, object oriented or s-expression oriented shells and domain specific languages to solve general problems like web development.

1. About Me

My name is Anthony DeDominic; I would dare call myself a professional GNU/Linux system administrator and developer with a broad range of experiences in the field of computer science. I come from Simi Valley, California, where I was born and raised. I have one brother, who is currently enlisted in the Army in the field of artillery. A fun fact about me: I don't use windows; I think this qualifies because it proves how hip and cool (or the complete opposite) I am by being the 1% that generally only uses Debian Linux.

2. Why Computer Science

I decided to study the field of computer science because I am fascinated with taking challenging and monotonous tasks and turning them into discrete and deterministic set of instructions. For instance, I'm currently studying with Dr. Tasneem, how to design and apply the concepts of domain specific languages to automate and solve continuous integration and continuous deployment problems; these problems, I've become to notice, are very tedious and time consuming to solve by hand.

I chose Eastern Connecticut State University to further my study due to the on-campus opportunities that I was offered by companies, like Cigna. I was glad of this as I met and learned many new concepts and had opportunities to collaborate with professors like Rosiene and Tasneem.

3. Future Plans

Currently my plans are to make a career out of free and open source software development; Ideally at a company like Red Hat, Software in the Public Interest, Google, et al. Free and open source software is my strongest passion. Outside of that, I may pursue a graduate program that allows me to specialize in linguistics; more specifically, linguistics focusing on computer science related problems. If possible, I would like to spend more time trying to solve hard problems around determinism and context-sensitive grammars.

4. Interests in Computer Science

As I've stated above, the two major things that interest me in computer science currently, are linguistics and free and open source software. My interest in linguistics primarily spawned from personal study parsing text. When I first started out, I had put a lot of faith into PCRE (Perl Compatible Regular Expressions). As I started getting more experience with them, I started to see the failings of just using PCRE (note: PCRE isn't really a regular language either, despite it's name). From there I started exposing myself to more advanced parsing concepts and techniques, such as parser generators (BNF, EBNF and PEG based), parser combinators and recursive decent parsers.

As I stated in my future goals, I care about linguistics and it's applications in computer science because one day I would like to make a computer that can process context-sensitive languages (think human languages!) *deterministically*, a feat that is currently, and potentially, unsolvable. Currently, to process human language or other complex, partially context-sensitive data, is to use concepts like graph-like neural-networks and machine learning. However, these solutions are not deterministic

and are thus not as pure and reproducible like a proven mathematical function.

5. Topics for Senior Research

Both of the ideas I will present here, fall in line with my interests of language studies.

5.1. Object Oriented Shells

The goal of this research is to develop Two new shells for Linux | POSIX systems which will have different syntax and shell builtins which attempt to bring a new form of structure on shell inputs and outputs.

5.1.1. Imperative Shells (bash)

Currently, using bash as an example (Borne Again SHell), one of the more popular shells for such systems, uses an old imperative design. Many of the tools that a bash shell would execute, including its own builtins, will accept and output text that isn't clearly structured using data structures like tables, maps, s-expression and so on. In order to indicate structure, bash, and many of the shell utilities it runs, depend on so called "arguments" which can be difficult to remember and must used precisely. Other variables, like IFS (Internal Field Separator) can dramatically change how your shell parses text. What results is a collection of scripts and shell utilities that do not necessarily conform to any particular structure and as a result, are not strictly portable.

5.1.2. Object Shells (powershell like)

One way to fix this is to introduce the concepts of "objects" to shells. Objects not only encapsulate data in easy, name indexed ways, but they also provide ways of manipulating or deriving data and other structures using encapsulated methods. The goal would be to have a shell which accepts and passes serialized data structures over the standard input and output stream of various shell utilities. The utilities will rely on standard defined classes to allow for easier comprehension of data.

5.1.3. Functional Shells (lisp like)

Another way is to require all shell interactions work on a tree-like structure called an s-expression. Lisp languages have been doing this for a long time. The benefits of this is that your code starts to become data as well, opening up interesting alleys for meta programming.

5.1.4. Goals of research

The following list is what the research should deliver.

1. Syntax - compare
   - comparison of these different shells and how they solve certain problems
   - comments on expressiveness of the varying styles
2. Performance - compare
   - attempt to understand potential performance impacts of adding structure to shell data
3. Develop
   - Two new shells, one that is OO and another that is lisp or functional like.

5.2. Domain Specific Languages

This would be more of a continuation of my research on solving CICD problems using domain specific languages. Specifically, looking into making a domain specific language to solve web development problems. This is challenging in the sense that web development problems are generally allow for more customization.

5.2.1. Goals

1. Scope
   - defining a clean, limited scope
   - verifying scope solves *MOST* web development problems
2. Comparisons
   - compare the simplicity of the domain specific language to popular web frameworks
   - like the javascript MEAN stack, PHP Laravel, Java Spring(Boot), etc.
   - determine if solving problems in DSL is more contrived than using a framework
3. Develop
   - a domain specific language for web problems with
   - a defined scope
   - compiler (lexer, parser, AST, target language)