Brian Christensen
Ben Nesbit
CS381 - Project 1 Proposal

This document details a simple project implemented in three languages. As a means to explore language fundamentals in multiple paradigms, Erlang and ML were selected for functional and Ruby for object-oriented. As an aside, regarding Erlang, its creator Joe Armstrong passed recently: Good night, sweet prince: https://en.wikipedia.org/wiki/Joe_Armstrong_(programmer)

The primary resources chosen include two popular books, *Learn You Some Erlang for Great Good!* by Fred Herbert (https://learnyousomeerlang.com/introduction) and *Introducing Erlang: Getting Started in Functional Programming* by Simon St. Laurent (https://github.com/oreillymedia/etudes-for-erlang). Additional resources include the Erlang documentation at http://erlang.org/doc/ and a linear algebra book by David Lay, *Linear Algebra and its Applications*.

The main features of Erlang include concurrency and resilience. This is done through recursion, message sending, pattern matching, and establishing pathways for data rather than telling the data where to go. Recursion and pattern matching are key aspects of functional programming paradigm, as well as single bound variables.

One big difference from ML is that it is impossible to rebind a variable in Erlang. If you say that x = 5, then if you try to say x = 6 there will be an error (x is NOT rebound), because 5 != 6.

“Variables in Erlang do not vary. You’re not supposed to program defensively. Processes are really, really, cheap, and you can have thousands of them or even millions, if you feel it. Oh, and then there is the strange syntax. Erlang doesn’t look like Java; there are no methods or classes and no objects. And wait a moment . . . even the equals sign doesn’t mean “equals” — it means “match this pattern.” (Herbert, XX)

Erlang bullets:

- Appeals to those looking for concurrency and parallelism
- Distributed computing
- Changing the value of any variable is strictly forbidden!
    - Like math

      y = 2
      x = y + 3
      x = 2 + 3
      x = 5
      then,
      x = 5 + 1
      x = x
      Therefore, 5 = 6 (this is dishonest and Erlang is an honest language)

- Referential transparency = concept of functions always returning the same result for the same parameter
- Actor model
    - Actors => processes
- Erlang is good at large software for server use (web servers, bidding and distributed database implementations, coupled with other languages, AI

The application is a kind of linear algebra calculator that will create a reduced row echelon form of a given system of linear equations. The echelon matrix is defined by the following properties:

- All nonzero rows are above any rows of all zeros.
- Each leading entry of a row is in a column to the right of the leading entry of the row above it.
- All entries in a column below a leading entry are zeros.

  If a matrix in echelon form satisfies the following additional conditions, then it is in reduced echelon form (or reduced row echelon form):

- The leading entry in each nonzero row is 1.
- Each leading 1 is the only nonzero entry in its column.

That this application requires use of lists, has a concrete algorithm (definition of the form), and requires manipulation of a variety of aspects of the lists makes this an ideal candidate to demonstrate the strengths and weaknesses of both functional and object oriented languages. Additionally, the applications of linear algebra are vast in the realm of computer science. This is not necessarily a strong reason to implement this for the context of the project, but perhaps is reason for intrigue.

I've seen things you people wouldn't believe. Attack ships on fire off the shoulder of Orion. I watched C-beams glitter in the dark near Tannhauser Gate. All these moments will be lost forever. Like tears in the rain.