Geno
A Programming Language for Computational Genomics

Project Proposal

COMS W4115 Summer 2014
Mohamed El-Refaey
mae2150@columbia.edu

June 2014
Abstract

Computational genomics is the domain concerned with the use of computational and statistical methods and techniques to decipher biology from genome sequences and related data (DNA and RNA sequence as well as other experimental data obtained with technologies that require the genome sequence, such as genomic DNA microarrays). With the current abundance of massive biological datasets, computational studies have become one of the most important means to biological discovery. The main task for scientists nowadays is to understand and analyze these sequences to decrypt the inner workings and secrets of the human body; and without the right analysis tools in hand, understanding and grasping the knowledge about genetics will be diminutive. Hence, the need for implementing a simple and efficient language that can express computational genomics data structure and enable the programmer to implement interesting computational genomics algorithms.

Geno is a domain-specific programming language that will be used to implement computational genomics algorithms (e.g. Reads Alignment algorithm etc.) and operations (Mutation, Sexual reproduction and recombination) as well as provide a standard library and data types that help makes the mission of biologist easy and enjoyable. It will also support basic programming language functionalities such as if-else statement blocks, loops, and variables.

Overview and Motivation

Currently, I am studying Computational Genomics course with Prof. Peer, and Programming Language and Compiler with Prof. Edwards. And I thought it will be invaluable experience to combine the knowledge I acquire in both subjects; so, I thought of designing a domain-specific language called ‘Geno’, which provide a means for Computational Genomics algorithms to be expressed, implemented and makes the genomics structure and operations as 1st class citizens while writing code in Geno. This will enable me to go deeply and thoroughly in genomics domain as well as the relevant computational algorithms used to understand the human Genome; and on the other hand will ignite good thoughts for language design and capabilities and what are the language constructs and data structure that will support Genomics. One extra motivation and a challenge in the same time: is to use this language in resolving the Computational Algorithms assignments and projects required in the Computational Genomics Course (Big goal that might not be feasible, as I study both subjects in same semester).

Language Constructs, Data Structure and Types

- Strings should be a 1st class citizen in Geno, as the DNA fundamental sequence representation is in the form of a string over $\sum = \{A, C, G, T\}$. Also RNA and amino acids as well as motifs are represented in stings and string patterns.
- Also, pattern matching should take a good portion of the Geno language design.
- The repetitive structure of the molecular strings requires a loop constructs in Geno (similar to for or while loops in other languages).
- Some computational genomics algorithms require the manipulation of DNA sequences in packs of bit vectors, and accordingly a bit-level operation is required.
- It might be good for Geno to support data structure like maps, hash tables, vectors, tuples, suffix trees, matrices (e.g. PAM, the 1st major Amino acid substitution matrix) etc.
- Standard library and functions: Geno should provide some standard library and built-in functions that help genetics; examples:
  o seqcmp() to compare to molecular sequences similar to strcmp in c++.
  o seqassemble() that help in aligning and merging fragments of of longer DNA sequence to reconstruct the original sequence.
- Math Operators (+, -, /, *, %)
- Relation and Comparison Operators(<, >, <=, >=, ==).
- Variables, the scope can be either global or local

**Design Questions and Food for Thoughts:**

Here are some questions popped up in mind when thinking of Geno language design that can be covered in the current Geno version or in future versions of the language:

- Shall Geno provide a way to build routines similar to what language like R provides? Or this should be left to the end programmer to devise his/her own statistical algorithms using primitive types that Geno provides?
- Shall this language provide a support to connect to backend DNA or molecular databases, and manipulate the database results in a certain way, or it will only be responsible of working with an in-memory online stream of molecular sequences? And also is there a way to support the searching of strings or substring Reads out of such databases? (The answer of this question is tightly related to the previous question whether to connect to database or not).
- How Geno will support a large-scale sequencing and sequence assembly especially with the huge amount chunks of DNA data available? And how memory and data structure in Geno manipulate this for better performance results for biologists programming with Geno?

**Computational Problem and motivating example**

Here is a motivating problem from the genomics world that can be solved using Geno:

**Align short reads to genome**

Input:
- Short Sequence Reads: l-long strings S1,..,Sm
- Reference genome: String R

Output:
- x1,..,xm along R where reads match, resp.
Challenges and Concerns

- Learning OCaml and Computational Genomics in same semester is a big challenge for me.
- Implementing a programming language for sensitive domain like genomics requires a precise and optimum way for handling genomics data structure and specially the strings representations of DNA sequences and base pairs (bp).
- Time may not be that enough to have a full fledged programming language for Computational Genomics (though I am really keen to do so!).