Variant representation, parser, generator, and coordinate converter

Contact Information:

Full name: Maximiliano David Bustos

E-mail: md.bustos90@gmail.com

Address: Vélez Sarsfield 156 – Torre 3 – 4C – Córdoba Capital - Argentina

Phone: +54 0351 4 231 750

Mobile phone: +549 351 6 313445

Skype: maxi_iclr

About me

My name is Maximiliano David Bustos, I am from La Rioja, Argentina. I am currently living in Córdoba, for study reasons.

I am 20 years old, and currently in the third year of Computer Science at "Facultad de Matemática, Astronomía y Física – Universidad Nacional de Córdoba".

About my programming experience:

- Started programming in C/C++ nine years ago.
- In 2006 I learned HTML, PHP and MySql and started developing web sites and web applications.
- I have participated in algorithm programming contests since 2003. I can mention: OIA, CIIC, TCHS and USACO (during High School); ACM-ICPC, TopCoder Algorithm Competitions, Google Code Jam (during university). I also participated in Mathematical Olympiads during High School.
- In 2008 I was selected to represent Argentina in the Intel International Science and Engineering Fair (Intel ISEF 2009). My work was titled *"School Timetable Generator Software"*. It was implemented using C/C++ and PHP.
- In 2011 I started using Python. I find this language quite easy to learn and I feel very comfortable with it. I am using it a lot at university for the subject Networks and Distributed System.
  Some weeks ago I participated in the PyCamp 2011 (a Python event), where I make some contributions to the Cdpedia Project, and started working in a new project: Pep8fy (a tool that makes your code more PEP8 compliant). Now I am a member of PyAr (the Python Argentina community).

I have a great interest in automata theory, languages and formal grammars (fields related to this project).

On the other hand, I have been interested in genetics since High School. The fact that small changes in some chemical structures might cause a great impact in human beings is very exciting for me. I think that this project is an opportunity to begin contributing to this field.

Abstract/Overview:

Computational analysis of genomic variation requires the ability to reliably translate between human and computer representations of genomic variants. While several standards for human variation syntax have been proposed, community support is limited because of the technical complexity of the proposals and the lack of software libraries that implement them. The goal of this project is to initiate freely-available, language-neutral tools to parse, generate, and convert between representations of genomic variation.

Expected contributions:

The main contributions this project will make to the community are:

- A formal grammar to clearly specify the HGVS nomenclature.
- A freely-available set of classes for representing and manipulating common types of genomic variation.
- A formatter and grammar-based parser for representing variations in the human-readable HGVS standard format.

Design aspects:

It will be important for the library to be flexible for two main reasons:

- We will need to add more variation representations later.
- Biology is a science that is constantly changing, so we will need to modify concepts that today we consider true.

The idea is to work at two levels: developing conceptual representation of many kinds of variants, and implementing a parser and formatter between HGVS nomenclature and this internal representation.

The library API will consist of:

- An abstract variant class and subclasses representing specific cases of them.
- Parsing and formatting methods.
- Syntactic and semantic checkers.
- Web services for coordinate conversion using NCBI Eutilities.

Implementation

I will implement the library in Python 2.7, using PLY (Python Lex-Yacc) or Pyparsing.

Proposal Timeline:

Before April 25:

- To familiarize myself with BioPython.
- Experiment with PLY (Python Lex-Yacc) and Pyparsing.
- Review some basic biology concepts: genomes, transcripts, proteins, genomic variation, and others suggested by my mentor.

- To familiarize myself with HGVS nomenclature, and experiment using Mutalyzer 2.0

Week 1 – April 25 - May 1:

- Define the subset of the HGVS to be represented.
- Identify variation types to be represented (SNV, CNV, repeats, inversions, etc).

Week 2 & 3 – May 2 – May 15:

- Design a formal grammar that fully represents the subset of the HGVS chosen before.
- Define a set of semantic validations for the represented variations.
- Write a short document explaining how to properly extend the formal grammar defined before.

Week 4 – May 16 – May 22:

- Set up a control version system and a bug tracker.
- Create a test suite for the library.

Week 5 – May 23 – May 29:

- ***Official coding period starts.***
- Library API specification.

Week 6 – May 30 – June 5:

- Library API design: classes, subclasses, methods, dependencies, exceptions and modules.

Week 7 – June 6 – June 12:

- Library API implementation: the abstract variant class and subclasses representing specific cases of them. Unit testing.

Week 8 & 9 – June 13 – June 26:

- Library API implementation: parser for the formal grammar defined before. Unit testing.

Week 10 – June 27 – July 3:

- Library API implementation: coordinate mapping between genomic, cDNA, and protein sequences. Unit testing

Week 11 – July 4 – July 10:

- First system testing instance.

Week 12 – July 11 – July 17:

- ***July 15: Midterm evaluation***
- Work on semantic validations. Unit testing.

Week 13 – July 18 – July 24:

- Implement web service for coordinate conversion using NCBI Eutilities. Unit testing.

Week 14 – July 25 – July 31:

- Implementation of a syntax checker as an example of the use of the library. Make it freely availably on the Internet.

Week 15 – August 1 – August 7:

- Second system testing instance.

Week 16 – August 8 – August 14:

- Documentation of the whole library.

Week 17 – August 15 – August 21:

- Integration of the library to BioPython.