Reliable Software in Bioinformatics: Sequence Alignment with Coq, Ada and SPARK

Karen Sargsyan

karsar@ibms.sinica.edu.tw

Abstract. Bioinformatics is becoming an indispensable tool for personalized medicine. Software is the most important factor to enable bioinformatics in practical applications and the reliability of such software has not been previously discussed. In this paper we share our strategy to ensure the reliability of the software for biological sequence alignment, which is the most widely used application in bioinformatics. We present the first findings of our project, which incorporates Coq, Ada and SPARK tools, and illustrate the reliability issues specific to bioinformatics.

1 Introduction

Bioinformatics is a computer science applied to biological problems, as a result, it is a subject to high expectations in the current era of computational technologies and advances in biology. Research in bioinformatics results in software, which means to provide solutions for biomedical problems. It is therefore of interest to discuss the reliability requirements which must apply to such software and to our knowledge, this is not currently taking place within the bioinformatics community. However, we do need highly reliable software in this field and we need to assess the requirements of its reliability.

The rapid development of bioinformatics during the last few decades brings huge advancements for the biomedical field, as well as new technical challenges for researchers. In the near future, doctors will have access to genetic data on which to base and tailor their medical treatment. To outline the advances in this field, we refer to the announcement of the full genome sequencing which costs less than $1000 [8]. In addition, a complete clinical assessment of a patient incorporating a personal genome [4] and the 1000 Genomes Project to sequence 1000 individuals [1] serve as further examples. Although these achievements merely appear to be of scientific value, recent developments benefit medicine directly. Thousands of DNA variants are associated with diseases and traits [14]. Chemotherapy medications (imatinib and trastuzumab) to treat specific cancers [11, 15] and a targeted pharmacogenetic dosing algorithm for warfarin [16, 17] demonstrate the potential of personalized medicine. Checking for susceptible genotypes for abacavir, carbamazepine and clozapine [13, 9, 7] reduces adverse effects. All of these examples demonstrate a rapid development in the field of personalized medicine. Reliability is important if life or health is dependent on the software in use. We expect the latter will become the case for bioinformatics

software in near future. Here, we pay attention to reliability issues, which specifically apply, to bioinformatics, whilst avoiding a discussion of measures that any reliable software development has to apply

2 Typical sources of errors in Bioinformatics

One of the biggest challenges in bioinformatics is the amount of data. It is not practical to store all results and make similar runs on the same datasets. These problems are still awaiting solutions and an increased speed of algorithms is desired. As a consequence, safety of the software is not a high priority for scientists in the short-term. However, we raise the importance of applying visible integration of bioinformatics with medicine and the time required to develop reliable software.

Errors come from different sources within bioinformatics. The error rate of available sequencing technologies results in significant challenges for applications. We still have to verify a novel DNA variant identification by placing it into its genomic context due to the high false positive rate. Verification itself is dependent on genome size and is time consuming. Therefore, it is not always a simple task to distinguish software errors from those which arise from data during experimental verification.

A programming language of choice can contribute to error accumulation in the code, if the language is not equipped with safety assurance tools or their usage is neglected. As illustrated in [10], popular choices of programming languages in bioinformatics, are C, C++, C#, Java, Perl and Python. Unfortunately, the only subjects of [10] are speed of execution and memory usage for popular algorithms in bioinformatics, with no analysis of software reliability provided. Moreover, R language for statistical software is omitted, although much is done in R to make statistical analysis reliable. In our survey we did not find the application of verification tools (provers, statistical analysis of the code, etc.), except for unit tests (which do not always exist) in conjunction with those languages for the case of bioinformatics software. The survey was conducted on published research articles and open source software, which are representative for the current stage of bioinformatics development. Furthermore, there are implementations of the new algorithms and verified/tested algorithms on a small number of datasets due to the absence of additional data. In severe cases, software is provided for the sole demonstration of the algorithm and verification of the results is provided in the paper, but with no further warranty provided for its accuracy. Despite these facts, several software tools have been developed over recent decades and the results of these applications prove their safety in scientific settings.

Application of the existing non-bioinformatics software for the analysis of biological data where no special software exists, may still lead to erroneous results. For example: Excel (Microsoft Corp., Redmond, WA) altered irreversibly gene names, which looked like dates [18].

3 A formal model of Sequence Alignment with Coq

Here we concentrate mostly on the Smith-Waterman (SW) local alignment algorithm. This algorithm may be briefly described as:

$$H(i,0) = 0, 0 \leq i \leq m$$

$$H(0,j) = 0, 0 \leq j \leq n$$

$$\text{If } a_i = b_j \rightarrow w(a_i, b_j) = w(match) \text{ or if } a_i \neq b_j \rightarrow w(a_i, b_j) = w(mismatch)$$

$$H(i,j) = max(0, H(i-1, j-1) + w(a_i, b_j), H(i-1, j) + w(a_i, -),$$

$$H(i, j-1) + w(-, b_j))$$

Where a,b are the strings over alphabet corresponding to nucleotides in case of DNA/RNA or amino-acids for proteins, m,n are the lengths of the sequences, $-$ denotes gap, $H(i,j)$- is the maximum similarity score and $w(x,y)$ is the scoring scheme. After obtaining $H(i,j)$, one starts with the highest value in it and moves backwards to one of positions $(i-1,j)$, $(i,j-1)$, and $(i-1,j-1)$, which has the highest value, until $(i-1,j)$ is reached. So $(i-1,j)$, $(i,j-1)$ moves correspond to adding gaps to the second and the first sequence accordingly.

The motivation for using local alignments is the existence of a reliable statistical model. Another incentive is difficulty in obtaining correct alignments in regions of low similarity for distantly related biological sequences. Whether local alignment as described above is a useful tool depends on how it agrees with biological experiment. Therefore, one may consider it as a formal model, which is the subject of experimental verification. In the context of software reliability, we need to note that the direct implementation of the algorithm will have long execution time, which is not fast enough for the majority of real-time applications. To solve this problem, more advanced implementations of SW are suggested [12, 2]. Their aim is to provide results similar to SW, with a faster performance in order of times. Because of the importance of SW, implementations using FPGA, CUDA and SIMD architectures exist. The next optimization of the algorithm is more complicated as it becomes harder to ensure the implementation is equivalent to the outcome of SW. This is where the proof assistants, such as Coq (http://coq.inria.fr), may be helpful. Availability of libraries, documentation and an organized community for Coq make it a natural choice for our project. Although, other alternatives to Coq may also work. Our approach is to implement a Coq module corresponding to SW and use it to prove equivalence to SW for each new optimized implementation. Coq also allows extraction of the proof in the form of a certified functional program.

These are arguments we would like to bring in justification of this strategy. It is always better to ensure the implementation used for medical purposes adheres to a well-known formal specification or a model. In scientific settings, it is clear which formal model is used and whether it sufficiently models reality, without guessing if the errors of implementation impact on observed disagreement. In

the case of such errors one either ignores the correct model or the errors mask the disagreement.

The complex biological problems have different models to achieve the same goals, with application under different conditions. For sequence alignment, the most widely used substitution of SW is BLAST [3]. BLAST is regarded as less precise than SW, producing inferior alignment under some settings, but having a practical execution speed. As its description is lengthy, we do not discuss it here. However, it is of interest to present a module specifying BLAST in Coq. This is valuable, since many optimizations of BLAST exist (CS-BLAST, CUDA-BLASTP, Tera-BLAST, etc.), similar to SW. Moreover, in such a setup, judgments about SW and BLAST are subject of exact proofs. To illustrate our ideas the author provides Coq codes for SW specification and an example of simple certified SW alignment program under the GPLv3 license (forbars.github.io). The listing of the code is not given here, as it is subject to further refinement.

It is possible to describe SW alignment in a language of paths on the graphs. Alternatively, one might choose another general mathematical framework in which SW alignment is a particular case. In our approach we do not assume any knowledge on the future form of formalized bioinformatics. Our first specifications are not of an abstract nature and our work follows an exploratory approach. In the author's view, it is risky to assume what the mathematics of bioinformatics will look like beforehand.

4 Reliable Sequence Alignment with Ada and SPARK

Although model specifications implemented in Coq are valuable in general, it is more interesting to apply specifications in a more practical realm. Our choice of language for a sequence alignment package implementation is Ada 2012 and SPARK 2014. Decades of development with attention on safety of code, compilation on a wide variety of platforms (including embedded applications) and acceptable speed of execution (in comparison to C) make these a natural option. It is not an easy task to modify a functional program extracted from Coq to the requirements of a particular (embedded and real-time) hardware. Therefore, our specifications in Coq are intended to formalize bioinformatics and do not deal with specific requirements, such as integer number representation in a provided system. SPARK, being a subset of Ada, allows proof of the correctness of subroutines before compilation. In addition, its subset is actively formalized in Coq [6], which makes it possible in the future to compare programs written in SPARK against specifications in Coq. Proving SPARK code with even fewer strict specifications is important, as testing of the bioinformatics tool involves large datasets and this makes it harder to prepare valid unit tests and find errors.

Ada contracts serve in a similar way, however, it is not always an option, as they check pre- and post- conditions during runtime. It is less desirable for medical software if a precompiled verification is available. Here, we avoid discussion of the improvements that are possible due to the access of suitable tools and methods in writing safe applications for general software case. It is worth noting,

the practices which exist for safe code have to be transferred to the software in bioinformatics. Rather, we present a specific issue in sequence alignment with demonstration of natural capabilities of Ada to provide a solution.

Different alignment algorithms often yield different results. This fact is widely ignored in practice. As a result of increased sizes of sequence datasets, faster alignment tools are needed. The trend is to make assumptions about the nature of frequent sequences and apply simplifications, which result in faster alignment, but this leads to a loss of quality. Methods incorporating sequence alignment as a subroutine are usually verified in experiments with a specific alignment in mind. Therefore, it is not immediately known how they behave if an alignment procedure is replaced with an alternative one. Moreover, those algorithms are dependent on parameters, such as weight matrix (BLOSSOM65 in BLAST). Weight matrices contribute to the difference between outputs, as their customized versions may be in use. Furthermore, tools for the sequence alignment are not always designed to provide output in the same format, as a consequence, several packages implement unification of interfaces for sequence alignment methods (an experiment in BioPython as an example). This makes it easy to interchange alignment tools in the code. However, as far as alignment algorithms are not equivalent, it is necessary to choose a suitable output and reject the production from the untested alignment with application in mind. This is where strong typing of Ada becomes important. Ada types are different in case their names differ, without accounting for the fact that the implementations are the same. As an example:

```
type Aligned is  ;
SW is new Aligned;
BLAST is new Aligned;
```

Here, we see two different versions of Aligned, which differ only in their name. However, functions and procedures that require the result of SW alignment will not accept the output of other forms of alignment. Thus, we associate separate types with outputs of different alignments. Strict distinction of the outputs, as proposed here, is not implemented or suggested in other packages/tools to our knowledge. It forces us to pay more attention to proven facts and methods in bioinformatics during software implementation and to avoid ignorance of the differences between sequence alignment algorithms.

5 Conclusion and Further Research

It has become even more apparent that bioinformatics software will play an important role in personalized medicine. However, current trends in software development for bioinformatics tend to give priority to the speed of the algorithms, together with complicated optimizations of existing implementations and accessibility, such as a service via cloud computing. Without denying the importance of all those topics, we raise the question of the implementation reliability of the core algorithms in bioinformatics and consider its importance for the future.

Our plans include providing Coq modules as formal specifications for the important algorithms in bioinformatics and the corresponding tools for developing bioinformatics software in Ada and SPARK. All specifications in Coq and some examples of SPARK and Ada code are published or are subject of publication under the GPL license (forbars.github.io).

References

[1] 1000 Genomes Project Consortium (2010), A map of human genome variation from population-scale sequencing. Nature, 467:1061-1073

[2] S. F. Altschul, B. W. Erickson (1986), Optimal sequence alignment using affine gap costs. Bulletin of Mathematical Biology 48: 603616.

[3] S. F. Altschul et al. (1990) Basic local alignment search tool. Journal of Molecular Biology 215 (3): 403410 (1990)

[4] E. A. Ashley, et al. (2010), Clinical assessment incorporating a personal genome. Lancet, 375:1525-1535

[5] D. A. Benson, et al. (2008), GenBank, Nucleic Acids Res. 36 (Database issue): D25D30

[6] P. Courtieu et al. (2013), Towards the formalization of SPARK 2014 semantics with explicit run-time checks using coq, HILT '13 Proceedings of the 2013 ACM SIGAda annual conference on High integrity language technology, pp: 21-22.

[7] M. Dettling, et al. (2007), Clozapine-induced agranulocytosis in schizophrenic Caucasians: confirming clues for associations with human leukocyte class I and II antigens. Pharmacogenomics J. 7:325-332

[8] R. Drmanac, et al. (2010), Human genome sequencing using unchained base reads on self-assembling DNA nanoarrays. Science, 327:78-81

[9] P. B. Ferrell, H. L. McLeod (2008), Carbamazepine, HLA-B*1502 and risk of Stevens-Johnson syndrome and toxic epidermal necrolysis: US FDA recommendations. Pharmacogenomics, 9:1543-1546

[10] M. Fourment, M. R. Gillings (2008), A comparison of common programming languages used in bioinformatics. BMC Bioinformatics. 9: 82 91.

[11] C. Gambacorti-Passerini (2008), Part I: Milestones in personalised medicineimatinib. Lancet Oncol., 9: 600

[12] O. Gotoh (1982), An improved algorithm for matching biological sequences. Journal of molecular biology 62(3):705708.

[13] S. Hetherington, et al. (2002), Genetic variations in HLA-B region and hypersensitivity reactions to abacavir. Lancet, 359:1121-1122

[14] L. A. Hindorff, et al. (2009), Potential etiologic and functional implications of genome-wide association loci for human diseases and traits. Proc. Natl Acad. Sci. USA, 106: 9362-9367

[15] C. A. Hudis (2007), Trastuzumabmechanism of action and use in clinical practice. N. Engl. J. Med. 357:39-51

[16] International Warfarin Pharmacogenetics Consortium et al (2009), Estimation of the warfarin dose with clinical and pharmacogenetic data. N. Engl. J. Med. 360:753-764

[17] H. Sagreiya, et al. (2010), Extending and evaluating a warfarin dosing algorithm that includes CYP4F2 and pooled rare variants of CYP2C9. Pharmacogenet. Genomics. 20:407-413

[18] B. Z. Zeeberg, et al. (2004), Mistaken Identifiers: Gene name errors can be introduced inadvertently when using Excel in bioinformatics. BMC Bioinformatics, 5:80-86