Logic-statistic modeling and analysis of biological sequence data: *A Research Agenda*

Henning Christiansen

Research group PLIS: Programming, Logic and Intelligent Systems
Department of Communication, Business and Information Technologies
Roskilde University, P.O.Box 260, DK-4000 Roskilde, Denmark
E-mail: henning@ruc.dk

Abstract. We describe here the intentions and plans of a newly started, funded research project in order to further the dialogue with the international research in the field. The purpose is to obtain experiences for realistic applications of flexible and powerful modeling tools that integrate logic and statistics, as exemplified by the PRISM system. As part of this, we will develop systematic and automatic optimizations, and the overall goal is to see how far it is possible to promote such techniques in computational biology.

1 Introduction

We describe the background and intentions of a newly started, funded research project. The primary project goal is to promote the application of logic-statistic modelling tools for the analysis of biological sequence data, where we use the PRISM system developed by Sato and Kameya [1, 2] as a starting point. Compared with traditional models based on HMMs and SCFGs, the techniques in consideration lift the expressive power to, in principle, Turing-complete languages. However, the price of the additional expressibility is comptuational complexity. In this project we will approach these computational problems with state-of-the-art program analysis and transformation techniques, including to see how existing and optimized implementations for traditional models can be integrated. This research is combined with investigations of biological problems which in themselves are relevant and provide good test cases. The project group include researchers in Computer Science and in Biochemistry from a number of unversities as well as industrial partners representing a major supplier of probiotic products for the dietary supplement industry, Chr. Hansen, plus a leading supplier of bioinformatics software, CLC bio; the project runs 2007–2011 and includes funding for several PhDs and postdocs, plus workshops etc.

There is an emerging trend in applying logic-based learning techniques for computational biology that we shall not review here, and in the present project we are especially interested in technologies based on logic programming as the central modeling paradigm. As is well known, logic programming is superior for modeling many linguistic phenomena due to elegance and flexibility, but not always with respect to efficiency.

We consider the abductive potential in such techniques. An overall model of the relationship between biological properties (understood in a very wide sense) and its encoding in a sequence (a protein, a genome, ...) is described as a model expressed as a statistic-logic program; known and already investigated sequences are used for training the model, i.e., for learning probabilities for its random variables; and with the learned probabilities, the model can be used for prediction of the "best" proposal for the biological properties encoded in a given sequence. This is quite analogous to the paradigm of linguistic discourse analysis as abduction [3] which we have studied in a logic programming context [4].

As test cases, we may use eukariotic problems from the literature for pressing complexity, but the main biological applications reflect the project group's expertise and practical needs in prokaryotic biochemistry. This include gene finding in health promoting bacteria, phylogenetic gene prediction, and prediction of gene function.

In section 2, we give an introduction to PRISM by a biologically inspired example which leads us to some observations why these techniques may be suited for biological sequence analysis. Section 3 describes our first nontrivial application for sequence data, which, however, does not involve prediction at its present state. Section 4 indicates some the computational problems that we intend to approach, and section 5 suggest possible extensions to the present PRISM, and section 6 provides a conclusion and discussion of a few specific points.

2 Working with sequence data in PRISM

Syntactically, PRISM extends the Prolog language with discrete, random variables, called `msw` for *multi-valued switch.* The following declarations, which are part of the model we describe below, define two singleton variables and two parameterized, and in principles, infinite classes of variables.

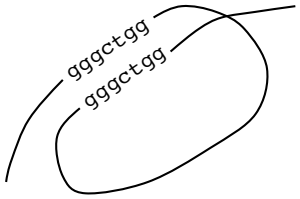
```
values(moreInBetween,[stop,continue]).
values(moreGlue,[stop,continue]).
values(which(_),[a,c,g,t]).
values(which(_,_),[a,c,g,t]).
```

Executing a call `msw(moreInBetween,X)` within a PRISM program will, by a random choice, assign one of `X=stop` or `X=continue`. As shown by [5], Prolog's traditional Herbrand model semantics generalizes immediately to a probabilistic semantics when probabilities are given for each random variable (provided that a few restrictions are respected on how `msw` is used in the program). In other words, such a PRISM program is a probabilistic model in the sense that it provides a probability distribution for all events that can be formulated in the program's logical language.

The following example program models a simple loop structure in a sequence; it is inspired by protein folding but apart from that it is completely artificial and has nothing to do with biology. If a sequence has the following form,

```
---------gggctgg-----------------------------gggctgg------
```

where a "-" indicates a random letter, a loop structure can arise assuming that each letter tend to glue to an identical letter as indicated by the following drawing.



Let us arbitrarily assume that the actual gluing zones can be relevantly described by a first order Markov model, and the stuff in between by a second order; in other words, we indicated that the two kinds of substrings are characterized by their own distribution of letters and other regularities. The following program defines a model for such sequences which captures these assumptions. Notice that random variables are used to generate the first copy of the gluing string, which then is repeated as a verbatim copy. Random variables `moreInBetween` and `moreGlue` determine the length of each subsequence. A goal `sequence(`G`,`S`)` means that sequence S has a loop glued together by the substring G.

```
sequence(G,S):-
    inBetween(-,S,S1),
    glue(G,-,-, S1,S2),
    inBetween(-,S2,S3),
    glueCopy(G, S3,S4),
    inBetween(-,S4,[]).

inBetween(L,S1,S2):- msw(moreInBetween,YN), inBetween2(L,S1,S2,YN).
inBetween2(_,S,S,stop).
inBetween2(L1,[L|S1],S2, continue):- msw(which(L1),L), inBetween(L,S1,S2).

glue(G,L1,L2,S1,S2):- msw(moreGlue,YN), glue2(G,L1,L2,S1,S2,YN).
glue2([],_,_,S,S,stop).
glue2([L|G],L1,L2,[L|S1],S2, continue):-
   msw(which(L1,L2),B), glue(G,L2,L,S1,S2).

glueCopy([],S,S).
glueCopy([L|G],[L|S1],S2):- glueCopy(G,S1,S2).
```

With fixed probabilities, this program can generate sets of samples which reflect the probabilistic semantics. PRISM can also learn probabilities from files of samples, which we can imagine represent sequences whose actual structure has been recorded in the laboratory. Assuming that the model is really appropriate for the sequences in questions, the detailed probabilities learned express regularities or structures within the substrings that may be new for the biological researcher, and in this way there is a flavour of induction. We can assume a file of training date as follows. The dominance of letters `g` and `c` in the gluing strings holds throughout the file; the use of bold face is for emphasis only and not part of the file.

```
sequence([g,g,g,c,t,g,g],[a,g,g,g,c,t,g,g,a,a,t,c,a,a,a,t,c,t,
        t,t,a,a,c,g,g,g,c,t,g,g,a,g,a,c,t,a,t,g,t,t,a,g,a,a,a,a]).
sequence(......, ......).
sequence(...., .....).
...
```

With the probabilities learned from these samples, PRISM can do Viterbi computations in order to predict, for a given sequence, the most likely glue substring out of the "logically" possible; this process can be compared with probabilistic abduction. See the following example, where the preferred gluing string (out of several possible) has similar high frequencies of `g` and `c` as the training data.

```
?- viterbig(sequence(G,[t,a,t,a,g,c,g,c,t,a,t,a,g,c,g,c,t,a,t,a])).
G = [g,c,g,c]
```

While being extremely simplistic, this example illustrates the qualities of PRISM that has motivated us to take it as initial platform for our research project:

- The clean semantics of the system ensures the we can accept a program as a formalized model of a phenomenon.
- Known probabilistic models, e.g., HMM, SCFG, Bayes' networks, etc. can be expressed in straightforward ways and, not least, *combined* while still maintaining a coherent model.
- Arbitrary data structures and auxiliary variables can be introduced whenever convenient without destroying the clean semantics.
- Compared with other linguistically oriented models, PRISM lifts to in principle Turing-complete languages (as opposed to regular (HMM) or context-free (SCFG)); the model above is based on a non-context-free language.
- The same model (= PRISM program) can be used for learning, sample generation, and prediction which gives a theoretically sound basis for comparing the different phases.[1]

3 Logic-statistic modeling for testing gene finders

Here we describe what seems to be the first non-trivial application of PRISM for a biological sequence problem. In a recent paper [6], we describe a partial model for genome sequences, which is used for the problem of producing artificial test data for testing existing gene finder programs; this problem is relevant since production of authentic test data is very expensive and sparsely available. The idea of using artificial test data for this purpose has been applied before [7, 8]. Basically, these works applied HMMs combined with ad hoc principles and it can be questioned [6] how "natural" the produced date are. Our aim of using PRISM is, of course, to provide a more detailed and reliable model, and thus better test data.

[1] Occationally it may be needed to use a few low-level Prolog hacks to have the program perform efficiently for the different phases, but done in a disciplined way, this may not destroy the semantics.

The model is defined for the intergenic subsequences capturing GC islands, occurrences of various sorts of repeat strings, and mutations. Probabilities were learned from the a set of sequences that were marked up using existing mapping tools. Since a full model was not available which also covers genes, we pasted together artificially produced intergenic subsequences with authentic genes. The conclusion from testing three different gene finders was that they all predicted too many genes and different genes. We shall not go into details here with this model, but only mention some distinct features of it that shows the flexibility a modeling tool which is based on a general and elegant programming language such as Prolog.

- Using PRISM's parameterized random variables, we defined a multi-level model, the top-level describing GC island, and embedding it in an abstract data type, each random variable and probability defined for the lower levels, exists in two versions for being in or outside of a GC island. The remaining part is a two-level Markov model, one level determining the alternation of different substring types (different sorts repeat strings, plain coloured noise, etc.) and the lowest level how these substring are built from letters.
  — This is a quite complicated model, but which by an experienced logic programmer can be organized in a readable way.
- A fairly large catalogue of named repeater strings were added, encoded from existing resources as a binary predicate; still this respects a reasonable semantics.
- To reduce complexity in handling mutations, we added a preprocessing phase for the training data, which uses a best match algorithm to produce exactly one detailed mark-up of the mutations that are supposed to have taken place when a (section of) a repeater string was copied into the sequence.
- Using a 64 bit version of PRISM we could, without any further optimizations, train from sequences of a total length up to a million in minutes.

However, it is important to notice that the problem of generating test data has been chosen carefully as our first exercise, as it is far less computationally complex than using the model for prediction of genes or other structures. We discuss consequences of this observation in the concluding section.

4 Computational problems considered in the project

A major problem in using PRISM "as is" for large sequences is storage consumption. First of all the sequences themselves which, when represented as Prolog lists, take up far too much space and, parodically, in the application described above we needed to upgrade to a 64 bit architecture, which means the data size doubles so that one letter in a sequence occupies 24 bytes as opposed to 1 byte or even down to 2 bits in a traditionally programmed architecture for sequence analysis. Secondly, PRISM uses an explanation graph which may grow to the size of the input data or worse; we would need to identify when a model or part thereof can do with simpler learning principles, e.g., simple counting in

some cases. There exist techniques for compact representation of trees which may inspire to similar methods for graphs.

An automatic analysis may identify when a model resembles, say a HMM, so that the system can switch to a specialized implementation. The project group includes developers for competitive HMM based software [9] so we have the possibility to interface with existing software.

Execution time may also become a problem since PRISM calculates all probabilities as correct as possible, which means that it will trace all possible explanations for a given observation, including those with such a small probability that it does not contribute in any significant way. The project group includes expertise in automatic program analysis and semantics preserving transformation of logic programs. However, for programs with a probabilistic semantics and a very high complexity, some sort of pruning is necessary. Thus, new techniques are in demand which produce not necessarily correct results, but results which are guaranteed not to deviate less than some epsilon from the true results.

One possible way to reduce complexity seems to be by splitting the sequence. Analysis of a sequence typically takes time which is a steep function of the length of the sequence. Splitting it into pieces analyzed separately and combining results is a way of reducing time consumption. Analysis of a model may indicate where "good" split points may be identified and what is the loss of precision.

Pre-processing and additional annotation of data. Models are often trained from previously marked-up sequences, and pre-processing the sequences by adding additional annotations can reduce complexity of the training phase. As an example, in the application described in section 3 and described in further detail in [6], we applied a best-match algorithm to produce unique descriptions of the mutation errors in occurrences of repeat strings. This technique made the training time roughly proportional to the sequence length. Such preprocessing mechanisms should be studied further and generalized.

5 Proposals for enhanced expressibility

We have identified until now two dimensions where it can be relevant with additional expressibility compared to PRISM.

The first one concerns numerical distributions. Using a random variable for each letter to determine whether to continue generating or stop, results in a geometric distribution of (sub-) sequence lengths, and this principle is inherent in Markov based models as well as our example in section 2. However, a geometric distribution gives a relatively high probability to short sequences which in many cases is not appropriate to characterize the lengths of substrings of a given kind. For the genomic sequence model described in section 3 we coped with this problem for GC islands by assuming a fixed minimum length followed by a tail generated by a geometric distribution; this is a hack also often applied with HMMs. This is a very inelegant solution and in most cases it does not give a reliable model. We consider extending with facilities so that normal distributions

or perhaps a sort of "generic smooth distribution" can be described and learned from observational data.

The second topic concerns constraints and is more difficult. A stochastic sequence model in PRISM (or an HMM or SCFG) typically describes a distribution of what is possible, but have difficulties in capturing that some of these strings are not desired. Consider, for example, the model for intergenic subsequences described in section 3; here the random choices may also capture patterns that represent genes by accident, so to speak. This may happen more frequently that one would expect since the repeat substrings that occur may contain (mutated and fragmented) remains of obsolete genes.

It can obviously be a problem for an application that generates data. Intuitively, a possible solution might be to analyze the sequences produced with a new model that captures undesired patterns so that they can be discarded. However, the theoretical justification of this approach may be problematic as it indicates subtle dependencies between the random variables of the model.

6 Conclusion and discussion

In this paper, we have presented a research project whose intention is to study how biological sequence analysis may be improved using advanced logic-statistic modeling and machine learning techniques based on logic programming. This involves both getting more experience in using and developing such models as well as improving the modeling facilities andtheir implementation.

At time of writing, the project is just about starting up, so we have only few and early results to report, obtained from an experiment in generating artificial test data for gene finding in eukariotic genomic sequences. There is a biological focus on prokaryotic genetics in the project, which means that sequences are measured in thousands as opposed to millions, so we may hope to get biologically interesting results even if not all computational problems that we identified, have been solved.

As we noticed in section 3, the task of generating artificial sequences seems computationally far less complex than the task of prediction, which is to identify the most probable structure of an observed sequence, For this reason, the models applied for gene prediction tend to tune down the level of sophistication; most gene finders are based on HMMs. The general lower complexity of data generation (including the preceding learning phase) means that it is appropriate in general to use much more fine-grained and hopefully more reliable models for this part. This may suggest a methodology for development of gene finders, starting from (a) a detailed model, as can be developed in PRISM or other tools of similar expressibility, and from it develop (b) a model biased towards efficient implementation. During this process (a) can be applied continually to test (b) and get an indication of how much precision is sacrificed. We may also hope that our project may lead to a reduction of the distance between models of the (a) and (b) types.

Acknowledgement: This work is supported by the project "Logic-statistic modelling and analysis of biological sequence data" funded by the NABIIT program under the Danish Strategic Research Council, and the CONTROL project, funded by Danish Natural Science Research Council.

References

1. Sato, T., Kameya, Y.: Statistical abduction with tabulation. In Kakas, A.C., Sadri, F., eds.: Computational Logic: Logic Programming and Beyond. Volume 2408 of Lecture Notes in Computer Science., Springer (2002) 567–587
2. Sato, T., Kameya, Y.: Parameter learning of logic programs for symbolic-statistical modeling. J. Artif. Intell. Res. (JAIR) **15** (2001) 391–454
3. Hobbs, J.R., Stickel, M.E., Appelt, D.E., Martin, P.A.: Interpretation as abduction. Artif. Intell. **63**(1-2) (1993) 69–142
4. Christiansen, H., Dahl, V.: Meaning in Context. In Dey, A., Kokinov, B., Leake, D., Turner, R., eds.: Proceedings of Fifth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT-05). Volume 3554 of Lecture Notes in Artificial Intelligence. (2005) 97–111
5. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP. (1995) 715–729
6. Christiansen, H., Dahmcke, C.M.: A machine learning approach to test data generation: A case study in evaluation of gene finders. In: International Conference on Machine Learning and Data Mining MLDM'2007, Leipzig/Germany. Lecture Notes in Artificial Intelligence, Springer (2007) To appear.
7. Burset, M., Guigó, R.: Evaluation of Gene Structure Prediction Programs. Genomics **34**(3) (1996) 353–367
8. Guigó, R., Agarwal, P., Abril, J.F., Burset, M., Fickett, J.W.: An Assessment of Gene Prediction Accuracy in Large DNA Sequences. Genome Res. **10**(10) (2000) 1631–1642
9. http://www.clcbio.com/.