EasyChair Preprint

№ 1822

The Propositional Logic of Nucleotide and Amino Acid Sequences

Pedro Cano and Ratilal Akabari

November 3, 2019

The propositional logic of nucleotide and amino acid sequences

Pedro Cano, MD, Ratilal Akabari, MS MB(ASCP)
Moffitt Cancer Center, Tampa, FL
12 Oct 2019

This is a study on the methodology to answer biological and clinical questions in terms of genetic information. We start from two basic assumptions: 1) The simplicity and efficiency of computer algorithms based on propositional logic offer a powerful robust solution to the problem of giving an account of biological and clinical facts in terms of genetic sequence data. And 2) Although genetic information is typically presented as character strings, it can also be presented as Boolean variables, which is a requirement for the use of propositional logic.

To our knowledge there are no previous attempts to design information models to manipulate DNA sequences with the tools of propositional logic by transforming nucleotide sequences into Boolean variables. Although we have used the method presented here for multiple applications in the past, it has never been presented in detail before.

1 Computer programmes = data structures + algorithms [1]

The algorithms that can be used to analyse DNA sequences depend of the data structures used to represent these DNA sequences. If DNA sequences are represented as strings of characters, then character string processing algorithms must be used. This is the reason why a language such as PERL has been so successful in bioinformatics.

Nevertheless, the use of other data structures to represent DNA sequences opens the door to other types of algorithms. This paper documents the efforts to develop alternative data structures for nucleotide sequences and amino acid sequences. (Whatever is said here about DNA nucleotide sequences also holds good for amino acid sequences.) These efforts aim to build the necessary information models to design efficient and reliable computer programmes.

2 Information models

In this study we limit ourselves to relational database models in order to maintain the rigour and efficiency of relational data integrity. A series of models of increasing complexity can be derived from each other in a consecutive chain. These models target the set of alleles of a particular gene. Our findings also apply to other related sets of nucleotide sequences or amino acid sequences.

2.1 Model 1

This is the standard basic model. Each record contains the sequence of one allele. A computer language capable of manipulating one or many character-strings like PERL, and algorithms like BLAST, are required.

In SQL one could use a function like substring (implemented in SQL Server), but with limitations and making SQL queries extremely complex.

Field name	Description	Example
PK	Primary key	123
FK_Allele	Foreign key to Allele table	HLA00001
Sequence	Nucleotide sequence	ccccagacg …

2.2 Model 2

As we design a more complex model, we simplify the necessary queries to answer our questions. Here a record contains a different column for each nucleotide position for a given allele. For a sequence of thousands of nucleotides, thousands of columns would be needed. Unless in the case of very short sequences, the increased complexity in table design is not justified. Nevertheless, we can restrict the nucleotide positions represented in the model to those that are polymorphic. This simplifies the table structure from having thousands of columns to having only a few dozens.

Field name	Description	Example
PK	Primary key	123
FK_Allele	Foreign key to Allele table	HLA00001
P_1	Nucleotide at position 1	c
P_2	Nucleotide at position 2	c
P_3	Nucleotide at position 3	c
…	One field for each position	

2.3 Model 3

The logical derivation of Model 2 is a table with one record for each nucleotide position for each allele. While Models 1 and 2 have one record per allele, here we have as many records for one allele as nucleotide positions.

XField name	Description	Example
PK	Primary key	123
FK_Allele	Foreign key to Allele table	HLA00001
FK_Position	Foreign key to Position table	1
ValueAtPositionN	Nucleotide at the corresponding position	a

Only polymorphic positions, where more than one nucleotide can be found, should be included. Conserved positions should be excluded. The difference between polymorphic and

conserved positions can be calculated by the entropy of that position, which is a sum of four terms, one for each nucleotide. A term being $[-f_n * \log(f_n)]$, where f_n is the frequency of nucleotide n at the position in question. When this entropy term is smaller than, let's say, 0.001, then the position in question is not polymorphic.

This model opens the door to powerful and simple SQL queries without the need of complicated character-string handling algorithms. One problem, however, is that for it to be successful, we need a reliable nucleotide position reference for the sequences of all alleles in a locus. This is a preliminary requirement in the implementation of this model.

2.4 Model 4

Model 3 can be developed into a new model in which the nucleotide at each position is represented as a Boolean variable. Each record in the table for this model has a field for the nucleotide position, another one for one of the four nucleotides, and another one for a Boolean value indicating whether it is true or not that a given nucleotide is present at a given position. This model allows the use of propositional logic algorithms, which are the most powerful algorithms that can be implemented with digital computers, in the sense that they are closest to the inner design and inner workings of digital computers.

Field name	Description	Example
PK	Primary key	123
FK_Allele	Foreign key to Allele table	HLA00001
FK_Position	Foreign key to Position table	1
Nucletide	Nucleotide	a
BooleanValue	Boolean value to indicate if the given nucleotide is present at the given position	True

All the possible nucleotide combinations for a given position must be included in the table for each allele. Obviously, for a given allele only one of the records for all the nucleotide combinations for a given position can have a True value. Like in Models 2 and 3, only polymorphic positions are included. Model 4 has four times as many records as Model 3.

2.5 Model 5

From Model 3 we can derive a number of table structures for different aspects of protein function. In the case of HLA, there are two main functional aspects: presentation of peptides to T cells, and the formation of the main histocompatibility antigens in transplantation.

Model 5 shows a record for each allele with the values as a series of amino acids for each antigenic determinant definition.

Field name	Description	Example
PK	Primary key	123
FK_Allele	Foreign key to Allele table	HLA00001
E_XX	Amino acids at epitope xx	X
E_XX	Amino acids at epitope xx	X
E_XX	Amino acids at epitope xx	X
…	One field for each epitope.	

2.6 Model 6

From Model 5 we derive Model 6 to give an account of histocompatibility antigenic determinants by using Boolean variables. In Model 6 we have as many records for each allele as there are antigenic definitions.

Field name	Description	Example
PK	Primary key	123
FK_Allele	Foreign key to Allele table	HLA00001
FK_EpitopeDefinition	Foreign key to EpitopeDefinition table	1
AminoAcids	Amino acids present at the given epitope	XXX
BooleanValue	Boolean value to indicate if the given amino acids are present at the given epitope definition	False

All the possible amino acid combinations for a given epitope must be included in the table for each allele. Obviously, for a given allele only one of the records for all the amino acid combinations for a given epitope definition can have a True value.

2.7 Other models

Just like models 5 and 6 are developed to account for the serologic properties of HLA molecules, other similar models can be developed to account for other functional characteristics of the protein in question, such as peptide presentation in the case of HLA.

3 Propositional logic

In these successive models not only the data structure changes, but also the data types change too. By transforming a character string into Boolean variables, propositional logic algorithms can be implemented.

Propositional logic is the calculus of Boolean variables, binary variables that can take one out of two possible values, like True / False, Yes / No, On / Off, 0 / 1, etc. As long as the things in the world can be seen as Boolean variables, then whatever we say of them is a propositional function. In this universe of Boolean variables, a scientific pursuit is the search for the most parsimonious

propositional function that accounts for the object under investigation, a dependent variable x (e.g. biological, clinical) in terms of a series of size n of independent variables x_1, x_2, ... x_n, such as the nucleotides at specific positions in the alleles of a gene.

3.1 Computational complexity of the problem

Although in a Boolean universe of n variables there are 2^n logically possible states, in reality this is constrained by the fact that what is logically possible may not be biologically feasible. So 2^n is just an upper boundary.

A collection of *n* variables results in a truth-table of 2^n rows. The number of possible truth-functions that can be defined on that truth-table is 2^*(2^n)*. If not all the variables need to be considered, then a different truth-table can be built for each of possible combinations –of any size– of the *n* atomic propositions. The number of possible truth-tables is therefore equal to the number of such combinations, which is equal to 2^n:

$${}_nC_n + {}_nC_{n-1} + {}_nC_{n-2} + \ldots + {}_nC_{n-n} = 2^n \qquad \text{[eq. 1]}$$

where ${}_nC_x$ is the number of combinations of size x taken from a collection of n elements.

For example:

number of variables	number of rows in a truth-table	number of possible truth functions in a truth-table	number of possible truth- tables
n	2^n	2^(2^n)	2^n
1	2	4	2
2	4	16	4
3	8	256	8
4	16	65536	16
5	32	4.2950 E+9	32
6	64	1.8447 E+19	64

If all possible truth-functions in all possible truth-tables need to be evaluated, the problem is very hard, of an order of complexity that prevents any practical solution. There are three sources of complexity: (I) the number of rows in a given truth-table, (II) the number of possible truth-function in a given truth-table, and (III) number of possible truth-tables.

(I) The actual number of rows to be included in the evaluation is limited, however, to those combinations of truth-values for which there are representative cases in the population of patients. If n = 300, the number of rows to evaluate is $2^{300} = 2E+90$, which is an absurd number of computational steps. But if there are, let us say, 1000 patients in our study population, there cannot be more than 1000 rows in the truth-table that needs to be evaluated, for combinations of truth-values of clinical variables without corresponding patients do not affect the calculations. In this way the number of rows in the truth table is reduced from 2^n to $\leq k$, where *k* is the size of the population of patients studied, or the

number of samples from patients, depending on the study and the clinical question being considered.

(II) The immense number of possible truth-functions (2^[2^n]) for a given truth-table of n variables and k rows is reduced to only one actualisation given by the empirical data which shows the value corresponding to the combination of variables in a given row. The complexity of finding the adequate truth-function in a given truth-table is solved empirically as we shall see in a minute. (See § 3.3)

(III) The question is now to see which is the optimal truth-table, or combination of atomic propositions that gives us the best answer to solve problem with the minimum number of variables. We intend to get rid of all variables that do not provide any additional information, either by itself or in combination with other variables, in the diagnosis of the disease under consideration. We have seen that the number of these combinations is 2^n. Is it possible to reduce this order of complexity?

3.2 An algorithm to search for the best propositional function

Because of the limitations in the complexity of the problem at hand, it is possible to devise a computer algorithm to search for the best propositional function that accounts for the clinical condition in question—or, rather, a likely best propositional function.

Let us define for each row ρ_i in a truth table

$$a_i = Card(\rho_i \wedge c) \qquad \text{[eq. 2]}$$

as the number of cases in the reference population *with* clinical condition *c*, where the clinical variables assume the truth-values represented for that row in the truth table; and

$$b_i = Card(\rho_i \wedge \neg c) \qquad \text{[eq.3]}$$

as the number of cases in the reference population *without* clinical condition *c*, where the clinical variables assume the truth-values represented for that row in the truth table.

We now define the *Z* value for each of the *m* rows of a truth-table:

$$Z = \frac{\sum_{i=1}^{m} \min(a_i, b_i)}{\sum_{i=1}^{m} (a_i + b_i)} \qquad \text{[eq.4]}$$

where, for each row in the truth-table:

$$\min(a_i, b_i) = \begin{Bmatrix} a_i, if\,(a_i \le b_i) \\ b_i, otherwise \end{Bmatrix} \qquad \text{[eq.5]}$$

For a given proposition ψ, the *Z* value is actually the number of false positives plus the number of false negatives over the total number of cases, or, in other words, the rate of misclassification.

Starting with all variables we evaluate one by one to see if it can be eliminated. That is, we create a new truth-table with the first variable removed from it. If *Z* does not increase, the variable removed in the evaluation cycle has no discriminatory power in the diagnosis of the clinical condition under investigation, either by itself or in combination with the rest of the variables and can be permanently eliminated.

The cost of maintaining a variable has to be measured against the value of decreasing *Z*. Therefore, the rule to eliminate a variable should be presented not as requiring *Z* not to increase at all, but rather not to increase beyond a level we call ζ which is the cost of maintaining a variable (e.g., the cost of performing a clinical test). If different variables have different maintenance costs, their elimination should be evaluated in relation to a different ζ value for each variable.

The elimination of one variable reduces the complexity of the problem from order $O(2^n)$ to $O(2^{n-1})$, therefore by proceeding in this fashion the problem can be solved in polynomial time rather than in exponential time. By evaluating the effect of the removal of one variable on the *Z* value, we not only evaluate a single combination of variables, but the $2^n/n$ combinations where that variable is involved.

The complexity of our problem is actually of order $O(k \cdot n)$.

What we have presented algorithmically can also be presented formally by saying that a clinical variable *v* can be eliminated from consideration if the simplification of a normal form of the Boolean function characterizing the clinical condition in question, either the disjunctive normal form or the conjunctive normal form, leads to the elimination of that variable. For example:

v_1	v_2	v_3	ψ
1	1	1	0
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

where proposition ψ has the disjunctive normal form:

$$f(v_1, v_2, v_3) = (v_1 \wedge \neg v_2 \wedge v_3) \vee (v_1 \wedge \neg v_2 \wedge \neg v_3) \qquad \text{[eq. 6]}$$

which can be simplified to:

$$f(v_1, v_2) = v_1 \wedge \neg v_2 \qquad \text{[eq. 7]}$$

reducing the function to only one term and eliminating variable v_3.

This only holds true if $\zeta = 0$. For higher values of ζ, only the algorithmic presentation will do.

3.3 A practical implementation of this algorithm.

The fundamental element in the practical implementation of this algorithm is the development of an evaluation function of a truth table based on empirical data.

Let ψ be a proposition function that accounts for clinical variable c in terms of a number of DNA-sequence properties, and let n_1, n_2 and n_3 be the DNA-sequence Boolean variables, each corresponding to whether the sequence has a given nucleotide at a given nucleotide position. (In a realistic experiment we would have many more than 3.) We have the following truth table with all the logical possibilities of the 3 Boolean variables, and the number of cases for each one of these combinations where clinical c is present, Card(c), and the number of cases for the same combination where c is absent, Card(¬c). If Card(c) > Card(¬c), we assign a True value (1) to ψ. If Card(c) < Card(¬c), we assign a False value (0) to ψ. Based on this assignment, we count the number of true positives (TP), false positives (FP), false negatives (FN) and true negatives (TN) for each row and then add them all together. These counts give us a 2x2 contingency table for each truth table. These 2x2 can be evaluated by means of the usual tools like the χ^2. This 2x2 contingency table also gives us the misclassification rate Z.

v_1	v_2	v_3	Card(c)	Card(¬c)	ψ	TP	FP	FN	TN
1	1	1	1	25	0			1	25
1	1	0	55	2	1	55	2		
1	0	1	1	2	0			1	2
1	0	0	1	45	0			1	45
0	1	1	0	3	0			0	3
0	1	0	15	3	1	15	3		
0	0	1	26	2	1	26	2		
0	0	0	0	0					
						96	**7**	**3**	**75**

We see that the last row for [$v_1 = 0$, $v_2 = 0$, $v_3 = 0$] has no actual empirical case and the logical space of 8 combinations is reduced to 7.

Although we formally described our heuristic algorithm by reducing one variable at a time from the original list of n variables, this can more easily be implemented as a computer programme working in then opposite way, adding one variable at a time:

1. Calculate the 2x2 contingency table and the corresponding χ^2 or any other such tools for each and every DNA-sequence Boolean variable in the original set of size n by itself. This will show us the best 1-variable ψ function. The variable selected we call v-I. This process has n steps.
2. Repeat process 1 to search for the best 2-variable ψ function where one of the variables is v-I. The second variable selected we call v-II. This process has n-1 steps.

3. Repeat the process to search for the best 3-variable ψ function where one of the variables is v-I and another one is v-II. The third variable in the function identified we call v-III. This process has n-2 steps.
4. Continue like this until the misclassification rate is no longer reduced beyond a preestablished threshold. This will give us a propositional function ψ with the optimal subset of the original set of Boolean variables that accounts for clinical variable c.

Of course, this approach is a heuristic approach and can lead us into a local minimum point without reaching the optimal solution which is at the overall minimum point. Variations of this algorithm can be easily implemented by exploring the surrounding solution space. For instance, by having several iterations of the algorithm in which instead of choosing the variable that gives us the best apparent answer in the first step, we choose the second, or other such alternatives, and we then compare the results of these alternative approaches. We must remember, however, that this is a heuristic approach to solve an intractable problem and we never have the assurance that we have reached the best solution, but just a solution that may be good enough for our purposes.

4 Applications

We have successfully used this approach for multiple studies to account for clinical and biological questions in terms of DNA nucleotide and protein amino acid sequences.

The method presented here was first used in combination with other mathematical models, geometric and algebraic in nature, to solve the following problems:

1. Prediction of which peptides are going to bind to which HLA alleles based on the amino acid sequences of the different HLA alleles. [2,3,4,5,6,7]
2. Analysis of gene expression array data in the diagnosis of leukaemia. [8]
3. Definition of the antigenic determinants that are the targets of anti-HLA antibodies. [9]

5 Conclusion

In our experience, formal methods that can be implemented by means of computer programmes are fundamental in medicine and biology. Their rigour provides a solid foundation to our understanding of clinical and biological problems. We eventually realised the computational superiority of the propositional logic approach over mathematical methods. This computational efficiency comes from the simplicity of propositional logic. The view of the world as facts that can be represented as logical (Boolean) variables has proven very powerful and applicable to a wide variety of problems. Incidentally, this view of the world comes from the philosophical work in logic by Frege, Russell and Wittgenstein, particularly from Wittgenstein's Tractatus Logico-Philosophicus.[10]

Galileo said:

> *Philosophy is written in this grand book, the universe, which stands continually open to our gaze. But the book cannot be understood unless one first learns to comprehend the language and read the letters in which it is composed. It is written in the language of mathematics, and its characters are triangles, circles, and other geometric figures without which it is humanly impossible to understand a single word of it; without these, one wanders about in a dark labyrinth.* [11]

Of course, the workings of the universe are not written in any such language, but one can *look at* the universe from the perspective of mathematics to much benefit. In order to understand the universe, we create mental representations: formal models. They can certainly be mathematical models, but they don't have to be, they can also be other type of formal models, like propositional logic, as in our case here. The formality of the models is what makes our epistemological enterprise scientific. And, for us, a formal model is one that can be implemented as a computer programme. We hold the position that our knowledge is only scientific if it can be represented by a computer programme whose output can be checked against our empirical observations.

6 References

1. Wirth N. Algorithms and data structures. Prentice Hall, Inc. 1986
2. Cano P, Fan B, Stass S. A geometric study of the amino acid sequence of class I HLA molecules. Immunogenetics 48(5):324-34, 1998.
3. Cano P, Fan B. A classification of HLA class I molecules in reference to their ability to bind peptides. Human Immunology 61:S57, 2000.
4. Cano P, Fan B. Effect of HLA class I polymorphism on peptide sequence requirements. Human Immunology 61:S57, 2000.
5. Cano P, Fan B. Similarities between peptide amino acid residues in their ability to bind to HLA class I molecules. Human Immunology 61:S58, 2000.
6. Cano P, Fan B. A geometric and algebraic view of MHC-peptide complexes and their binding properties. BMC Struct Biol 1:2, 2001.
7. Cano P. The propositional calculus of peptide binding to the major histocompatibility complex. In: Immunobiology of the Human MHC. Proceedings of the 13th International Histocompatibility Workshop and Conference. II. Ed(s) JA Hansen. IHWG Press: Seattle, WA, 265-270, 2006
8. Cano P. The propositional calculus of gene-expression array data. Journal of Computational Biology 7:634, 2000.
9. Cano P, Fernandez-Vina M. Two sequence dimorphisms of DPB1 define the immunodominant serologic epitopes of HLA-DP. Human Immunol 70(10):836-43, 2009.
10. Wittgenstein L. Tractatus Logico-Philosophicus. Routeledge and Kegan Paul Ltd. 1961 Translated by D F Pears and B F McGuinness
11. Galileo Galilei. The Assayer. 1623. Translated by Stillman Drake