Decision Diagrams for Qualitative Biological Models

Michel Le Borgne, Philippe Veber

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Decision Diagrams for qualitative biological models.

Michel Le Borgne, Philippe Veber

N˚6182

May 2007

Systèmes biologiques

ISSN 0249-6399 ISRN INRIA/RR--6182--FR+ENG

Decision Diagrams for qualitative biological models.

Michel Le Borgne, Philippe Veber

Systèmes biologiques
Projet Symbiose

Rapport de recherche n˚6182 — May 2007 — 17 pages

Abstract: The modelisation of large biological systems such that metabolic networks or gene interaction networks, implies interpretations of heterogeneous biological knownledge. A common features of observations accumulated by biologist on these systems its their qualitative nature. More over it often happens that the knownledge is based on comparisons between different experimental conditions. In previous papers variations in observed variables between two conditions was interpreted in terms of equilibria shift. This leads to use qualitative equations in a sign algebra. The present report gives details on the implementation and main algorithms involved.

Key-words: qualitative algebra, bio , static models

(Résumé : tsvp)

Unité de recherche INRIA Rennes
IRISA, Campus universitaire de Beaulieu, 35042 RENNES Cedex (France)
Téléphone : 02 99 84 71 00 - International : +33 2 99 84 71 00
Télécopie : 02 99 84 71 71 - International : +33 2 99 84 71 71

Diagrammes de Décision pour des Modèles Biologiques Qualitatifs.

Résumé : La modélisation de grand reseaux biologiques (métaboliques, géniques ou mixtes) demande d'interpréter de nombreuses connaissances hétérogènes accumulées par les biologistes. En ce qui concerne le comportement de ces réseaux, ces informations sont le plus souvent de nature qualitative. De plus, beaucoup d'observations portent sur une comparaison entre une ou plusieurs situations expérimentales où les systèmes biologiques peuvent être considérés en équilibre. Dans plusieurs papiers précédents, nous avons montré que les variations observées entre deux situations expérimentales, pouvaient être considérées comme résultant de déplacements d'équilibre. Les signes de ces variations sont soumises à des contraintes dans l'algèbre des signes qui permettent de confronter, dans une certaine mesure, modèles qualitatifs et expérimentations. Dans ce rapport nous présentons notre implémentation des équations qualitatives comme des polynomes sur des corps finis. Des algorithmes spécifiques sont décrits en détail.

Mots-clé : algèbre qualitative, bioinformatique, modèles statiques

1 Qualitative variational models in integrative biology

Several families of mathematical models of biological network have been proposed. One of the oldest one is the family of models based on differential equations. These models are rooted in the studies of the kinetic of individual reactions in biochemistry. The differential model of a network is obtained by composing differential equations modeling individual reaction, assuming additivity of metabolite fluxes. Extensions have been proposed to gene activation and signalisation.

1.1 Models derived from differential equations

Despite the number of studies devoted to enzyme activity, only a a few reaction kinetic constant values are known. Moreover, the measurements of kinetic constants are in general made in vitro and it is not established that the values are the same in vivo. In [8], [10] a qualitative model was proposed which overcome to some extend, the lack of quantitative data. Let us recall briefly the derivation of such a model.

We consider a network of interacting cellular constituents. These products may be proteins, RNA transcripts or metabolites for instance. The state vector X denotes the concentration of products in the system. X is assumed to evolve according to the following differential equation:

$$\frac{dX}{dt} = F(X) \tag{1}$$

where F is an (unknown) non linear function. A steady state X_{eq} of the system is a solution of the algebraic equation

$$F(X_{eq}) = 0 \tag{2}$$

An experiment on the system consist in a comparison between two situations of interest and is modelized as an equilibrium shift. As a result to a change in the parameters (experimental settings), the system goes from its initial stable steady state to another one. In the end, we observe ,the sign of the total variation in concentration for a subset of products. Our aim here is to understand how variation signs are constrained by the interactions between products.

Let us denote $s(j,i) = \mathrm{sgn}(\frac{\partial F_i}{\partial X_j})$ where F_i is the ith component of F. We assume that $s(i,i) = -$, i.e. the diagonal of the Jacobian matrix of F are strictly negative. For a detailed discussion of this hypothesis see [8]. Moreover the sign of each entry $\frac{\partial F_i}{\partial X_j}$ of the jacobian matrix is assumed to be constant if this entry is not null. Taking the total derivative at an equilibrium point, we get:

$$dX_i = -\left(\frac{\partial F_i}{\partial X_i}\right)^{-1}\left(\sum_{k\in \mathrm{pred}(i)} \frac{\partial F_i}{\partial X_k} dX_k\right) \tag{3}$$

Under mild conditions discussed in [8], it is possible to derive an equation on signs of concentration variations between two equilibria points:

$$s(\Delta X_i) \approx \sum_{k\in pred(i)} s(k,i)s(\Delta X_k) \tag{4}$$

where $s(\Delta X_k)$ denotes the sign of the total variation of the concentration of kth chemical specie. Notice that these sign equations are valid not only for small variations but also for fairly large ones.

These equations in the sign algebra $\{+,-,?\}$ where **?** stands for an indeterminate sign, constitutes a model for qualitative variations. It is comparable with some models used in economics for comparative static analysis.

1.2 Models given in terms of influences

It often happens that biological knownledge is directly expressed in term of influences. Assertions like " when specie A concentration increases then specie B concentration decreases" are very common and some data bases such as

RegulonDB([7]) collect such type of knownledge. This type of observation is very common in gene interaction description.

It may be hypothesized that this graph results from the observation of the variations between two equilibria of a differential model. This kind of knownledge is often visualised as a graph named an interaction graph. Nodes represents molecules and edges represents relations between molecules. In this case, edges are labeled by signs and we get an signed influence interaction graph.

A major difference with qualitative models derived from differential equation, is the composition law of influences on one component (specie concentration, gene activity ...) of the model. Although a simple additive rule may be encountered, in many cases, the composition of influences targetting a component is a complex boolean function and varies from one chemical specie to another one. Think for example of a trascription factor which might induce the expression of a gene g_1 and repress the expression of another gene g_2.

Nevertheless, gathering this kind of informations on a set of biological component ended up in a set of constraints on a set of qualitative quantities. Very often these quantities belongs to a sign algebra $\{+,-,?\}$ but in some cases several levels of concentrations are considered. The technique presented in this report applies also to these more complex qualitative algebras. For a brief presentation of sign algebra see [9], [5].

1.3 A qualitative dynamical model

A qualitative dynamical model has been proposed for gene networks [4]. This model is based on the assomption that gene expressions can be modelled by a piecewise constant linear differential equation. The bounded domain of concentration vectors is partitioned with a finite number of hypercubes. On each hypercube, the coefficients of the linear model are constant. Moreover, in this equation, all the components are independant.

For each gene, the hypercube partition induces a concentration domain partition by intervals. Each interval corresponds to some level in the concentration of the gene product. So qualitative values can be associated to these levels and the evolution of the levels of gene expression is given by a family of relations, one for each gene g:

$$F_g(X, x'_g) \tag{5}$$

Where X is the present vector of gene expression levels, x'_g is the next level of expression of gene g. It must be mentioned that given an expression level of the gene, there are several admissible next expression level for each gene. So the dynamical system is intrinsiquely non deterministic.

Although this report is devoted to static models, we would like to mention that the encoding of qualitative systems on finite fields can be used for studying dynamical systems. In particular model checking can be implemented in this coding and was already used for circuit verification and real time program verification [6].

2 Finite fields and the coding of qualitative equations

Finite fields are of widespread use in computer science. There are used in circuit verification, coding, cryptography and many other domains. Their use in circuit verification was mad possible by the discovery of an efficient representation of polynomial functions on the field $\mathbb{Z}/_{2\mathbb{Z}}$ [3]. This representation using a data structure known as BDD (binary decision diagrams), is based on the well known Shannon decomposition of boolean functions. In [] a similar representation of polynomial functions on $\mathbb{Z}/_{3\mathbb{Z}} = \{0, 1, -1\}$ was proposed and used for the verification of signal processing programs.

2.1 Finite fields

Finite fields or Galois fields are fields with a finite number of elements. The number of elements is necessary of the form p^n where p is a prime. For a quick look at finite field theory see [2]. A Galois field of order p^n is traditionaly denoted $\mathbf{GF}(p^n)$.

For representing qualitative values, it is necessary to chose a Galois field with order in accordance with the finite number of values taken by each variables. In this report we will be mostly interested by simple sign algebra where each variable take value in the set $\{+,-,?\}$. This is in relation with our biological application and all the algorithms presented here are implemented in this case. In this setting, the Galois field $\mathbb{Z}/3\mathbb{Z} = \mathbf{GF}(3^1)$ is our favorite.

2.2 Polynomial functions on finite fields

Assuming a finite field K being choosen to represent the qualitative values of interest, then every constraint, every evolution equation on the qualitative values are described by finite families of functions from K^n into K. Due to the finiteness of the field, we will see soon that every such function tuns out to be a polynomial function.

The ring of polynomial functions in several variables on a finite field K of order k is itself finite. It doesn't coincide with the ring of polynomials which is infinite. It is also a finite vector space on the field K.

Let $X = (x_1, ..., x_n)$ a set of n variables and α an element of K. The α-Lagrange polynomial in x_i is:

$$L_\alpha(x_i) = \frac{\prod_{\beta \in K; \beta \neq \alpha}(x_i - \beta)}{\prod_{\beta \in K; \beta \neq \alpha}(\alpha - \beta)} \tag{6}$$

The Lagrange polynomials satisfy: $L_\alpha(\beta) = 0$ if $\alpha \neq \beta$ and $L_\alpha(\alpha) = 1$.

The family of polynomials

$$L_{(\alpha_1,...,\alpha_n)} = L_{\alpha_1}(x_1) \times \ldots \times L_{\alpha_n}(x_n) \tag{7}$$

for all $(\alpha_1, \ldots, \alpha_n)$ in K^n is a basis of the vector space of polynomial functions with n variables on K.

As a consequence, it is easy to show that every function with several arguments on a finite field K is a polynomial function. If f is defined on K^n with value in K, we get:

$$f(x_1, \ldots, x_n) = \sum_{(\alpha_1,...,\alpha_n) \in K^n} f(\alpha_1, \ldots, \alpha_n) L_{\alpha_1}(x_1) \times \ldots \times L_{\alpha_n}(x_n) \tag{8}$$

This equality is a direct consequence of the properties of Lagrange polynomials.

It is also possible to generalize Shannon decomposition of boolean functions to polynomial functions on finite field. Given f a (polynomial) function on K^n, let us denote $f|_\alpha(x_2, \ldots, x_n)$ the polynomial function on K^{n-1} resulting from the substitution of the first variable x_1 by the value α. Notice that this definition assumes an order on the variables.

It is then straightforward to check the generalisation of the Shannon decomposition:

$$f(x_1, \ldots, x_n) = \sum_{\alpha \in K} L_\alpha(x_1) f|_\alpha(x_2, \ldots, x_n) \tag{9}$$

If at least two polynomial functions $f|_\alpha$ are different, x_1 is called the rootvar of f and denoted $rootvar(f)$. If the three functions $f|_\alpha$ are all equal, then f is independant from x_1.

Shannon decomposition is the starting point to the representation of polynomial functions on finite fields as Direct Acyclic Graphs (DAG). The first step is to derive a tree representation from the generelized Shannon decomposition by applying it recursively. The second step follows from the remark that it often happends that many subtrees are identical. Then eliminate this redondancy by representing each subtree only once. Figure 1 illustrates the process for $K = \mathbb{Z}/3\mathbb{Z}$. For the original idea due to Bryant, have a look at Wikipedia [1].

2.3 Coding of qualitative equations

Our goal is to implement efficient computations on qualitative algebra. For illustrating the method we will use the sign algebra $\{+,-,?\}$. Application of the method to other finite qualitative algebra is similar.

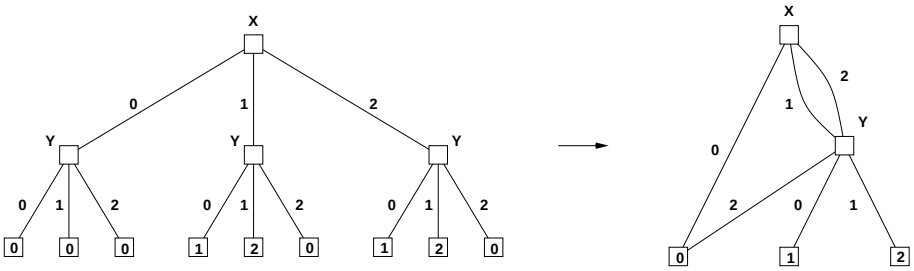

Figure 1: From tree representation to direct acyclic graph for $X^2(Y+1)$. The tree has 13 nodes while the DAG representing the same function has 5 nodes.

A sign algebra comes not only with symbols such that **+**, **−**, **?** but also with operations on symbols defining a specific calculus. The sign algebra is endowed with two composition laws: qualitative sum and qualitative product.There are specified by the two tables:

$\oplus$	**+**	**−**	**?**
+	**+**	**?**	**?**
−	**?**	**−**	**?**
?	**?**	**?**	**?**

$\otimes$	**+**	**−**	**?**
+	**+**	**−**	**?**
−	**−**	**+**	**?**
?	**?**	**?**	**?**

In addition, sign algebra has a relation, improperly named equality, denoted as $\approx$ and defined as follows:

$\approx$	**+**	**−**	**?**
+	T	F	T
−	F	T	T
?	T	T	T

It has special properties. In particular the qualitative equality is not an equivalence relation, since it is not transitive. This implies that computations in qualitative algebra must be carried with care. The two major properties to be emphasised from the above tables are:

- if a term of a sum is indeterminate (**?**) then the whole sum is indeterminate.
- if one hand of a qualitative equality is indeterminate, then the equality is satisfied whatever the value of the other hand is.

A solution of a qualitative system is a vector without indeterminate component satisfying the equations.

The coding We code **+** as 1, **−** as -1 and **?** as 0. In order to code qualitative equations, we need to code the qualitative sum and the qualitative product. A straightforward check using value enumeration shows that the qualitative sum $X \oplus Y$ can be coded as the polynomial function $\mathbf{sq}(X,Y) \stackrel{\text{def}}{=} XY(X+Y)$ while the qualitative product $X \otimes Y$ is simply code by the product of polynomial functions. Qualitative equations $X \approx Y$ are translated into polynomial equations following the pattern $\mathbf{eq}(f,g) \stackrel{\text{def}}{=} fg(f-g)$. Solutions are obtained by decoding solutions of the polynomial equation $eq(f,g) = 0$.

In fact the implementation of the qualitative equations doesn't use the sum and product of polynomial functions. We take advantage of very usefull property of Shannon decomposition. Given Δ any binary operation defined on polynomial functions, then we have:

$$f_1 \Delta f_2 = \sum_{\alpha \in K} L_\alpha(x_1) f_1|_\alpha \Delta f_2|_\alpha \tag{10}$$

So to implement a binary operation it is sufficient to have a generic implementation of binary operations and a table for the constants. Qualitative operations are implemented with the tables 2.3. With this coding, every qualitative system has a solution if and only if the corresponding polynomial system has a solution without null component. Null solutions are excluded since **?** solutions are excluded for qualitative systems. In general we will have to add polynomial equations $X_i^2 = 1$ for each variable X_i, to insure this condition.

2.4 Former approach of qualitative equations

Sign algebra have allready been used in qualitative modeling applied to various plants enginered by humans. Several heuristics were proposed for the resolution of qualitative systems. For linear systems, set of rules have been designed [5]. This set is complete: it allows for finding every solution. It is also sound: every solution found by applying these rules is correct. The rules are based on an adaptation of Gaussian elimination. But a systematic use of elimination is not possible as it is the case for linear systems over a field. Instead, only heuristics exists for choosing the equation and the rule to apply on it. In case of dead-end when no more rule can apply, it is necessary to backtrack to the last decision made.

So, programs implementing qualitative resolution were not very efficient in general and only problems of small size can be resolved in reasonable time. Our approach, inspired by circuit verification techniques, enlarge the range of application of qualitative algebra.

3 Problems from biological models

We have briefly presented in section 1 how static qualitative models may be used for modeling biological systems. In the present state of biological experimental technologies, it is possible to have qualitative measurements on a great number of variables. For exemples, microarray techniques allow for the simultaneous observation of the activity of thousand of genes. The challenge for bioinformaticians is to integrate numerous data from various experiments together with previous knownledge on the biological system of interest. One way to integrate biological previous knonledge is to build a static qualitative model.

Moreover, new experimentation techniques known under the generic name of hight throuput techniques became recently available in molecular biologie. They allow for simultaneous observations of many variables. With the coding of qualitative equations as polynomial functions, we have a new technique to deal with models which scale with the new technologies.

3.1 Simple problems

Consistency Given a model (a set of qualitative equation $\mathscr{E}$), the first question coming to mind is: is my model coherent?. A necessary condition for this is to have non trivial solutions. That is, in the qualitative domain, at least one solution as defined previously. A solution is obtained by defining a function encoding the conjunction of equations. In $\mathbb{Z}/_{3\mathbb{Z}}$ such a function is, for two equations P, Q:

$$f(P,Q) = (P^2 + Q^2)^2 \tag{11}$$

since $f(P,Q) = 0$ if and only if $P = 0$ and $Q = 0$. This is easily proven by simple enumeration or with the help of Fermat little theorem. For any finite field a similar function is defined by a table such that $f(\alpha,\beta) = 1$ if $(\alpha,\beta) \neq (0,0)$

and $f(0,0) = 0$. Applying such a function repetively replaces a set or system of equations by a unique equation

$$f(p_1, f(p_2, \ldots, f(p_{n_1}, p_n)\ldots)) \tag{12}$$

with the same set of solutions.[1]

So checking the consistency of a static qualitative model turns out in checking wether this unique equation is the constant polynomial function 1 or not. If it is 1, there is no solution and the model is not consistent. Recall that we add the equations $X_i^2 = 1$ to the model equations $\mathscr{E}$) to eliminate undeterminate solutions.

Many practical questions are solved in the same way such that the compatibility of a model with experimental data or the efficiency of an experiment. All these questions are about the set of solutions of the model.

3.2 Prediction

One of the main role of models in science is to allow for the computation of predictions. When qualitative models are in use, the notion of prediction from the model is not quite clear. We cannot consider every solution of the system of equations as a prediction. Consider a biological system modeled as a static model on sign algebra with each variable x representing the variation of a chemical specie concentration. If the model has solutions with $x = +$ and solutions with $x = -$ this translates to x may increase or decrease. This can hardly be considered as a prediction!. This motivates the definition of hard components.

A component x of a vector X is a hard component for the system of qualitative equations $\mathscr{E}$ if for each solution $(\alpha_1, \ldots, x = \alpha, \ldots, \alpha_n)$ of $\mathscr{E}$, x takes the same value.

Hard components is a well known concept in qualitative modeling. It has no counterpart in polynomial function theory. So we had to develop an original algorithm to compute hard components of a polynomial function.

3.3 Experimental error

It may happen, and for a qualitative model it is the most interesting case, that experimental data do not fit the model. More precisely, the model obtained by substituing experimental data for the corresponding variables in the model $\mathscr{E}$ is not consistent (i.e. has no solution). In such a situation it may be either that the model is wrong, either some experimental data are questionable.

Computations based on the model cannot solve this issue. Its a matter of scientific methodology. But computations can help to analyse this falsification of the model by putting under focus some variables which are more questionnable than others. If we are rather interested in model fitting, it could be usefull to know on what parameters to focus in order to make the minimal changes in the model when it does not agree with experimental data.

A solution to this issue can be given by considering the solutions of the model which are the closest to the experimental data. In discrete spaces as sign algebra cartesian products, the only interesting distance is the Hamming distance. Recall that the Hamming distance between two vectors $(\alpha_1, \ldots, \alpha_n)$ and $(\beta_1, \ldots, \beta_n)$ is the number of components such that $\alpha_i \neq \beta_i$. So the Hamming distance between experimental data and solutions of $\mathscr{E}$ gives the minimal number of data which could be modified to fit the model. If variables represent parameters of the model, the Hamming distance gives the minimal number of adjustment of the model to fit the experimental data.

Of cause, it is not sufficient to have the minimum number of corrections. We are also interested in the closest vectors fitting the model and in the sets of minimal corrections. Let $d_h(v_1, v_2)$ denote the Hamming distance between v_1 and v_2 two vectors of the same dimension on the sign algebra. If E is the set of solutions of $\mathscr{E}$ we define:

$$d_h(v, E) = min_{w \in E}(d_h(v, w)) \tag{13}$$

The closest vector from v fitting the model is the Hamming projection of v onto E:

$$p_h(v, E) = \{w / \ w \in E;\ d_h(v, w) = d_h(v, E)\} \tag{14}$$

[1] In the implementation we use a dichotomic scheme which is equivalent but much more effi cient

and the set of minimal corrections is:

$$corr_h(v,E) = \{u/\ v+u \in E\ \&\ d_h(v,v+u) = d_h(v,E)\} \tag{15}$$

4 Computation of hard components

Computation of hard components is not a standard operation on polynomial functions on finite fields. However, it is possible using, Shannon decomposition as in standard operations. Let $X = (x,X')$ be a vector in $(\mathbb{Z}/_{3\mathbb{Z}})^n$ where the first component x corresponds to the first variable for the order used for QDD representation. The remaining components are components of the vector X'. If

$$f(x,X') = L_\alpha(0)f|_0(X') + L_\alpha(1)f|_1(X') + L_\alpha(-1)f|_{-1}(X') \tag{16}$$

is the Shannon decomposition of f then the computation of hard components is based on a few remarks:

- (x,X') is a solution of $f(X) = 0$ if and only if X' is a solution of $f|_x(X') = 0$
- x is a hard solution of $f(x,X) = 0$ if and only if there is only one value α for which $f|_\alpha$ has roots.
- a component x_i of X' is a hard component for f if and only if it is a hard component with the same value in every function $f|_\alpha$ having roots..

The representation of QDD by directed acyclic graphs result in large gains in memory compared to a raw representation as a tree derived from Shannon decomposition. However, all computations on QDD are based on this decomposition and thus potentialy have an exponential time complexity. The gain in memory is based on redondancy of many subtrees in the decomposition. This means that many computations are repeated if the algorithms are implemented by brute force application of Shannon decomposition. A fairly standard way to avoid repeated computations is to partially memorize them in a cache and this is used in the QDD package.

A technical problem with a cache is that it is generaly dedicated to cache one type of objects. The main cache of the QDD package is dedicated to QDD memorisation. We could build a cache for each type of object or some kind of universal cache. This would add a lot of complexity to the software. In our implementation we chosed to represent hard components as a polynomial function having the vector of hard components as unique root in the space of its variables. In this context, by variables of a polynomial, we mean variables appearing explicitely in the QDD representation.

Let $X = (x,X')$ and p' is a polynomial function representing hard components among the variables of X'. If x is a new hard component to be represented with value α, then a polynomial function representing the hard components of p' and α is:

$$p = L_\alpha(x) \times p'(X') \ + \sum_{\beta \neq \alpha} L_\beta \times 1 \tag{17}$$

With this in mind, it is easy to compute efficiently hard components of a polynomial function f considered as an equation. If

$$f(x,X') = \sum_{\alpha \in K} L_\alpha(x) f|_\alpha(X') \tag{18}$$

is the Shannon decomposition of f then the preceding remarks apply:

- if none of the $f|\alpha$ has roots then f has no roots and consequently no hard component.
- if only one of the $f|\alpha$ has roots, then x is a hard component with value the corresponding α. The hard components among the variables of X' are the hard components of $f|\alpha$.
- if several of the $f|\alpha$ have roots, then x is not a hard component and hard components of f are the hard components common to the $f|\alpha$ having roots and sharing the same value.

5 Hamming algorithms

In computing Hamming distance, projection and correction, we need to consider a more general case than the introductory one in 3.3. In general it is impossible to observe all the variables of a model. So the vector of observations V is indexed by a subset of the set of variables $X = (X_1, \ldots, X_n)$. Let us introduce:

$$var(V) = \{X_i / i \text{ is a component index of } V\} \tag{19}$$

If the observations involve only a subset of variables, the Hamming distance from the set of solutions of $\mathscr{E}$ represented by a unique equation $f(X)$ is redefined as:

$$d_h(V, f) = min\{d_h(V,W) / var(V) = var(W) \ \& \ \exists Y \ f(V,Y) = 0\} \tag{20}$$

It is the distance from V to the projection of the solutions of $f(X) = 0$ to $var(V)$.

Hamming correction definition has also to be made more precise:

$$corr_h(V, f) = \{U / \ var(V) = var(U) \ \& \ \exists Y \ f(V+U,Y) = 0 \ \& \ d_h(V,V+U) = d_h(V, f)\} \tag{21}$$

The Hamming projection is adapted in the same way;

$$proj_h(V) = \{U / \ var(V) = var(U) \ \& \ \exists Y \ f(U,Y) = 0 \ \& \ d_h(V,U) = d_h(V, f)\} \tag{22}$$

.

In order to establish the recursive equations for the computation of the Hamming distance, Hamming projection and Hamming correction, we asssume that the variables are ordered and the components of V are ordered with the same order. By isolating the first variable we can write $X = (x, X')$ and $V = [v, V']^t$ where exponent t denotes the transpose of a vector or a matrix. In the Shannon decomposition, we use the same order on variables:

$$f(x, X') = L_\alpha(0) f|_0(X') + L_\alpha(1) f|_1(X') + L_\alpha(-1) f|_{-1}(X') \tag{23}$$

where at least two of the polynomial functions $f|_\alpha$ are distinct so that x is the rootvar of f.

The variable associated with the first component v of V is not necessarily equal to x. It might be equal, before or after x for the order on the variables. These three cases are the three cases of the recursive equation.

Identities on Hamming distance We first establish simple identities on Hamming distance. Le E be a set of solutions of a system of qualitative equations $\mathscr{E}$ and $W = [w, W']^t$ a vector in E. If $V = [v, V']^t$ is a vector with $var(v) = var(w)$, the following identity holds:

$$d_h(V,W) \ = \ d_h(v,w) \ + \ d_h(V',W') \tag{24}$$

If *true* is coded by 1 and $false$ by 0, $d_h(v,w) = (v \neq w)$. Taking the minimal value on W we get:

$$\begin{aligned} min_{W \in E} d_h(V,W) &= min_{W \in E}(d_h(v,w) + d_h(V',W')) & (25)\\ &= min_{w \in E_{var(w)}}(d_h(v,w) + min_{W' \in E'_w} d_h(V',W')) & (26)\end{aligned}$$

where

$$\begin{aligned} E_{var(w)} &= \{w / \exists W' \ s.t. \ (w,W') \in E\} & (27)\\ E'_w &= \{W' / (w,W') \in E\} & (28)\end{aligned}$$

The first set is the projection of E onto the w component and the second one is the fiber over w. It is possible to get rid of the first set by setting $min_{W' \in \emptyset} d_h(V',W') = \infty$. With this convention we get:

$$min_{W \in E} d_h(V,W) = min_{w=0,1,-1}(d_h(v,w) + min_{W' \in E'_w} d_h(V',W')) \tag{29}$$

5.1 Case 1: $var(v) = rootvar(f)$

Distance computation Applying the preceding identities on the set $E = \{W / \exists Y \ f(W,Y) = 0\}$ we get:

$$E'_w = \{W' / \exists Y \ f(w,W',Y) = 0\} \quad (30)$$
$$= \{W' / \exists Y \ f|_w(W',Y) = 0\} \quad (31)$$

and with $d_h(V,E) = d_h(V,f)$ when E is the set of solutions of the equation $f(X) = 0$ we can write:

$$min_{W' \in E'_w} d_h(V',W') = d_h(V', f|_w) \quad (32)$$

which leads to the recursive equation:

$$d_h(V,f) = min_{w=0,1,-1}(d_h(v,w) + d_h(V', f|_w) \quad (33)$$

With the convention on empty sets we get the end cases for the recursion: $d_h(V,1) = d_h(V,-1) = \infty$ and $d_h(V,0) = 0$.

Projection computation We assume that $d_h(V,f)$ is computed. Let $rh(V,f) = \{\alpha / d_h(V,f) = d_h(v,\alpha) + d_h(V', f|_\alpha)\}$ and W an element of $proj_h(V,f)$. we get $d_h(V,f) = d_h(v,w) + d_h(V',W')$. Now if Y is such that $f(W,Y) = 0$, by Shannon decomposition we have:

$$f(W,Y) \ = \ f|_w(W',Y) \ = \ 0 \quad (34)$$

Moreover, $d_h(V', f|_w) = d_h(V',W')$ since if W'_1 were a vector such that there exist Y_1 with $f|_w(W'_1,Y_1) = 0$ and $d_h(V',W'_1) < d_h(V',W')$, then $W_1 = [w,W'_1]^t$ satisfies $f(W_1,Y_1) = 0$ and $d_h(V,W_1) < d_h(V,W)$, showing that W is not in the Hamming projection $proj_h(V,f)$. We conclude that $w \in rh(V,f)$ and $W' \in proj_h(V', f|_w)$.

Reciproquely, if $w \in rh(V,f)$ and $W' \in proj_h(V', f|_w)$ then the vector $W = [w,W']^t$ satisfies:

$$d_h(V,W) \ = \ d_h(v,w) \ + \ d_h(V',W') \ = \ d_h(v,w) \ + \ d_h(V', f|_w) \ = \ d_h(V,f) \quad (35)$$

If we denote in the same way a polynomial having $proj_h(V,f)$ as solutions, we can write down a recursive equation on polynomial functions:

$$proj_h(V,f) \ = \ \sum_{w \in rh(V,f)} L_w \times proj_h(V', f|_w) \ + \ \sum_{not \ w \in rh(V,f)} L_w \times 1 \quad (36)$$

Correction computation The computation of the Hamming correction derives directly from the previous equation. We look for vectors U such that $V+U$ is in $proj_h(V,f)$. The preceding equation applied to $U+V$ gives:

$$proj_h(V,f)(U+V) \ = \ \sum_{w \in rh(V,f)} L_w(u+v) \times proj_h(V', f|_w)(U'+V') \ + \ \sum_{not \ w \in rh(V,f)} L_w(u+v) \times 1 \quad (37)$$

which is equal to zero if and only if $u+v \in rh(V,f)$ and $proj_h(V', f|_w)(U'+V') = 0$ for $w = u+v$. This shows that U' is in $corr_h(V', f|_{w-v})$ and using the identity $L_w(u+v) = L_{w-v}(u)$ we get:

$$corr_h(V,f)(U) \ = \ \sum_{w \in rh(V,f)} L_{w-v}(u) \times corr_h(V', f|_{w-v})(U') \ + \ \sum_{not \ w \in rh(V,f)} L_{w-v}(u) \times 1 \quad (38)$$

again by using the same notation for a set and a polynomial equation representing this set.

5.2 Case 2: $var(v) > rootvar(f)$

Projection computation In this case, the polynomial function doesn't put any constraint on the components of vectors with index $var(v)$. In other words, in the Shannon decomposition of f, with $var(v)$ as first index, $f|_0 = f|_1 = f|_{-1}$. In the Hamming distance expression $d_h(V,f) = min_w(d_h(v,w) + d_h(V',f|_w)$, it is possible to have $w = v$ and we get:

$$d_h(V,f) \;=\; d_h(V',f|_v) \;=\; d_h(V',f) \tag{39}$$

As an immediate consequence $proj_h(V,f) \;=\; proj_h(V',f)$.

Correction computation As the first component of vector V belongs to a vector in the Hamming projection, the correction on this component is $u = 0$. For the other part U' of the vector U, we get from preceding relation:

$$d_h(V+U,f) \;=\; d_h(V'+U',f|_v) \;=\; d_h(V'+U',f) \tag{40}$$

showing that U' is in $corr_h(V',f|_v)$. From these remarks we get for the polynomial function:

$$corr_h(V,f)(U) \;=\; L_0(u) \times corr_h(V',f) \;+\; L_1(u) \times 1 \;+\; L_{-1}(u) \times 1 \tag{41}$$

5.3 Case 3: $var(v) < rootvar(f)$

This case includes the case $V = \emptyset$ which is not used directly but appears as an end case in recursion equations. We will first establish the recursive equations for $V \neq \emptyset$.

Distance computation For computing the distance we start from the very definition (20):

$$\begin{aligned}
d_h(V,f) &= \min_Y\{d_h(V,Y)/\ f(Y) = 0\} && (42)\\
&= \min_{Y'}\{d_h(V,Y')/\ \exists y\ f(y,Y') = 0\} && (43)\\
&= \min_{Y'}\{d_h(V,Y')/\ \prod_{y=0,1,-1} f|_y(Y') = 0\} && (44)\\
&= d_h(V, \prod_{y=0,1,-1} f|_y(Y')) && (45)\\
&= d_h(V', \prod_{y=0,1,-1} f|_y(Y')) && (46)\\
&= \min_{y=0,1,-1} d_h(V',f|_y) && (47)
\end{aligned}$$

since the set of solutions of $\prod_{y=0,1,-1} f|_y(Y'))$ is the union of the sets of solution of each factor and $d_h(V',f|_y) = \infty$ whenever $f|_y$ has no root.

Projection computation From the preceding equations, if W satisfies $d_h(V,f) = d_h(V,W)$ then there exists a Y' and a y such that $f|_y(W,Y') = 0$ and the *min* is reached for y. So $y \in rh(V,f)$ and W is necessarely a vector of $proj_h(V,f|_y)$.

Reciproquely if $y \in rh(f,V)$ and $W \in proj_h(V,f|_y)$ then there exist a Y' such that $f|_y(W,Y') = 0$. This vector satisfies the property: $f(W,[y,Y']^t) = f|_y(W,Y') = 0$. Since V has no variable corresponding to the index of $rootvar(f)$ which is the index of the component y, we get:

$$d_h(V,W) = d_h(V,f|_y) = min_{y=0,1,-1} d_h(V,f|_y) = d_h(V,f) \tag{48}$$

since $y \in rh(V,f)$.

so we get: $proj_h(V,f) = \cup_{y \in rh(V,f)} proj_h(V,f|_y)$ and in polynomial function form:

$$proj_h(V,f) = \prod_{y \in rh(V,f)} proj_h(V,f|_y) \tag{49}$$

Correction computation The computation of the correction derives directly from the computation of the projection. If $V+U \in proj_h(V,f) = \cup_{y \in rh(V,f)} proj_h(V,f|_y)$ then there exists a $y \in rh(V,f)$ and $V+U \in proj_h(V,f|_y)$. Consequently, $U \in corr_h(V,f|_y)$. The converse inclusion is obvious and we get:

$$corr_h(V,f) = \prod_{y \in rh(V,f)} corr_h(V,f|_y) \tag{50}$$

as for the projection.

Case when $V = \emptyset$ Since this case appears only as an end case in recursive equations, let us consider the last step of a recursion ending with $V = \emptyset$. At this point we must have:

$$\begin{aligned} f &= L_0(x)f|_0(X') + L_1(x)f|_1(X') + L--1(x)f|-1(X') && (51)\\ V &= [v] && (52)\end{aligned}$$

and $var(V) = \{x\}$. On one hand we get from the recursive equation:

$$\begin{aligned} d_h(V,f) &= min_{y=0,1,-1}(d_h(v,y) + d_h(\emptyset, f|_y)) && (53)\\ &= min\{d_h(\emptyset,f|_v),\ 1+d_h(\emptyset,f|_{v_1}),\ 1+d_h(\emptyset,f|_{v_2})\} && (54)\end{aligned}$$

where v_1, v_2 are the two other values in $\mathbb{Z}/3\mathbb{Z}$ different from v.

On the other hand we have from our Hamming distance definition:

$$\begin{aligned} d_h(V,f) &= min_{y=0,1,-1}\{d_h(v,y) / \exists Y'\ f|_y(Y') = 0\} && (55)\\ &= min_{f|_y \text{has roots}} = (d_h(v,y)) && (56)\\ & && (57)\end{aligned}$$

The only way to conciliate the two equations is to set:

$$\begin{aligned} d_h(\emptyset, f|_\alpha) &= 0 \text{ if } f|_\alpha \text{)has roots} && (58)\\ d_h(\emptyset, f|_\alpha) &= \infty \text{ either} && (59)\end{aligned}$$

Projection computation Since equation 36 must apply in the last step of the recursion, we are interested in the values which are in $rh(V,f)$. From equation (54) two cases appears:

- f_v **has roots** Then v is the only element in $rh(V,f)$ and from $proj_h(V,f) = L_v(x)proj(\emptyset, f_v) + \sum_{w \neq v} L_w(x)1$ we must have $proj_h(, f_v) = 0$
- f_v **has no root** Then the elements of $rh(V,f)$ are the $w \neq v$ such that f_w has roots. Since $proj_h(V,f) = L_v(x)proj(\emptyset, f_v) + \sum_{w \neq v} proj(\emptyset, f_w)$ we must have $proj_h(\emptyset, f_v) = 1$ and $proj_h(\emptyset, f_w) = 0$ when f_w has roots and 1 if not.

To summerize:

$$\begin{aligned} proj_h(\emptyset, f) &= 0 \text{ if f has roots} && (60)\\ &= 1 \text{ either} && (61)\end{aligned}$$

Correction computation Correction computation follows the same path. Using the same argument and starting from equation 38 and the expression of the distance (54) we get:

$$corr_h(\emptyset, f) = 0 \text{ if } f \text{ has roots} \quad (62)$$
$$= 1 \text{ either} \quad (63)$$

6 Random solution of a qualitative equation

In some applications we need to compute the proportion of solutions of a qualitative equation in a given reference space. Linked to this problem, is those of selecting a solution at random with a uniform distribution. If n is the number of solutions of a qualitative equation represented by a QDD f, we want to select a solution $(x_1,\ldots,x_k)$ with probability $\frac{1}{n}$. From Bayes's rule we get:

$$p(x_1,\ldots,x_k) = p(x_2,\ldots,x_k/x_1)p(x_1) \quad (64)$$

Let n_α the number of solutions of $f|_\alpha$. Since $(x_2,\ldots,x_k)$ is a solution of $f|_\alpha$ drawn with uniform probability among all the solutions of $f|_\alpha$, we get:

$$p(x_1 = \alpha) = \frac{n_\alpha}{n} \quad (65)$$

Let $p = n/3^k$ the proportion of solutions of the equation f in the space corresponding to the variables $(X_1,\ldots,X_k)$ and $p_\alpha = n_\alpha/3^{(k-1)}$ the proportion of solutions of $f|_\alpha$ in the corresponding space. It is easy to establish the relation $p = (p_0+p_1+p_2)/3$ from which we derive:

$$p(x1 = \alpha) = \frac{p_\alpha}{p_0+p_1+p_2} \quad (66)$$

An algorithm may be builded on the preceding relations: first compute for each node of the QDD representing f, the proportion of solutions in the corresponding space. Then traverse this graph, down from the root, selecting each branch corresponding to the value α, with the probability given by (66). Unfortunately, such an algorithm would mean memorizing one float per node of the QDD, which may be quite hudge. To avoid this problem we propose to build a solution and compute the proportion of solutions at each node during the same traversal. The solution will be coded with the same trick we used for hard componants. This allows to use the general cache to memorize computations. Assume also that we have a function $Random(\pi_0,\pi_1,\pi_2)$ which delivers a value α in the range $\{0,1,2\}$ with the probabilities given as arguments.

Let p_α be the proportion of solutions of $f|_\alpha$ and sol_α the random solution (coded as a QDD) drawn for $f|\alpha$. If

$$f(x_1,\ldots,x_n) = \sum_{\alpha\in K} L_\alpha(x_1) f|_\alpha(x_2,\ldots,x_n) \quad (67)$$

is the Shannon decomposition of f, and $ps = p_0+p_1+p_2$, we can draw $\alpha = Random(p_0/ps, p_1/ps, p_2/ps)$ and then we get the recursive relations:

$$sol = L_\alpha sol_\alpha + \sum_{\beta\neq\alpha} L_\beta \quad (68)$$
$$p = ps/3 \quad (69)$$

Stopping cases are obvious. Remark that a complete implementation of this algorithm must take into account the case where some variables associated with dimensions of the ambient space may not appear in the QDD f. This needs slight adaptations of the algorithm we will not describe here.

7 Conclusion

We have presented original algorithms on polynomial functions on finite fields. These algorithms arised in the application of finite field to qualitative theory in order to provide efficient computation means for biological applications. The algorithms have been implemented as a library written in C programming language. Python binding are also available providing easy access to these algorithms. The Python module, named Pyquali, has been sucessfully used in checking the coherence of data bases of gene interactions in E.Coli. A work under development uses the same kind of computation to infer a qualitative static model of gene interactions in Saccharomyces cerevisiae.

References

[1] Wikipedia: Binary decision diagrams, http://en.wikipedia.org/wiki/binary_decision_diagram.

[2] Wikipedia: Finite fields, http://en.wikipedia.org/wiki/finite_field.

[3] R.E Bryan. Graph-based algorithm for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, August 1986.

[4] H de Jong, JL Gouzé, C. Hernandez, Page M., T. SARI, and Geiselmann J. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin of Mathmatica Biology*, 66:301–340, 2004.

[5] J.L. Dormoy. Controlling qualitative resolution. In *Proceedings of the seventh National Conference on Artificial Intelligence, AAAI88', Saint-Paul, Minn.*, 1988.

[6] H. Marchand, E. Rutten, M. Le Borgne, and M. Samaan. Formal verification of programs specified with signal : Application to a power transformer station controller. *Science of Computer Programming*, 41(1):85–104, August 2001.

[7] H Salgado, S Gama-Castro, M Peralta-Gil, E Diaz-Peredo, F Sanchez-Solano, A Santos-Zavaleta, I Martinez-Flores, V Jimenez-Jacinto, C Bonavides-Martinez, J Segura-Salazar, A Martinez-Antonio, and J Collado-Vides. RegulonDB (version 5.0): Escherichia coli K-12 transcriptional regulatory network, operon organization, and growth conditions. *Nucleic Acids Res*, 1(34):D394–7, Jan 2006.

[8] A. Siegel, O. Radulescu, M. Le Borgne, P. Veber, J. Ouy, and S. Lagarrigue. Qualitative analysis of the relation between dna microarray data and behavioral models of regulation networks. *Biosystems*, 84:153–174, 2006.

[9] L. Travé-Massuyès and P. Dague, editors. *Modèles et raisonnements qualitatifs*. Hermes sciences, 2003.

[10] P Veber, M Le Borgne, A Siegel, S Lagarrigue, and O Radulescu. Complex qualitative models in biology: A new approach. *Complexus*, 2(3-4):140 – 151, 2004.

Contents

Unité de recherche INRIA Lorraine, Technopôle de Nancy-Brabois, Campus scientifique,
615 rue du Jardin Botanique, BP 101, 54600 VILLERS LÈS NANCY
Unité de recherche INRIA Rennes, Irisa, Campus universitaire de Beaulieu, 35042 RENNES Cedex
Unité de recherche INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 MONTBONNOT ST MARTIN
Unité de recherche INRIA Rocquencourt, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex
Unité de recherche INRIA Sophia-Antipolis, 2004 route des Lucioles, BP 93, 06902 SOPHIA-ANTIPOLIS Cedex

Éditeur
INRIA, Domaine de Voluceau, Rocquencourt, BP 105, 78153 LE CHESNAY Cedex (France)
http://www.inria.fr
ISSN 0249-6399