BIOINFORMATICS ORIGINAL PAPER Vol. 26 no. 13 2010, pages 1637–1643
doi:10.1093/bioinformatics/btq240

Systems biology Advance Access publication May 6, 2010

Polynomial algebra of discrete models in systems biology

Alan Veliz-Cuba[1], Abdul Salam Jarrah[1,2] and Reinhard Laubenbacher[1,*]

[1]Virginia Bioinformatics Institute, Department of Mathematics, Virginia Polytechnic Institute and State University, Blacksburg, VA, USA and [2]Department of Mathematics and Statistics, American University of Sharjah, Sharjah, UAE

Associate Editor: John Quackenbush

ABSTRACT

Motivation: An increasing number of discrete mathematical models are being published in Systems Biology, ranging from Boolean network models to logical models and Petri nets. They are used to model a variety of biochemical networks, such as metabolic networks, gene regulatory networks and signal transduction networks. There is increasing evidence that such models can capture key dynamic features of biological networks and can be used successfully for hypothesis generation.

Results: This article provides a unified framework that can aid the mathematical analysis of Boolean network models, logical models and Petri nets. They can be represented as polynomial dynamical systems, which allows the use of a variety of mathematical tools from computer algebra for their analysis. Algorithms are presented for the translation into polynomial dynamical systems. Examples are given of how polynomial algebra can be used for the model analysis.

Contact: alanavc@vt.edu

Supplementary information: Supplementary data are available at *Bioinformatics* online.

Received on January 29, 2010; revised on April 25, 2010; accepted on April 28, 2010

1 INTRODUCTION

Finite dynamical systems, that is, discrete dynamical systems with a finite state space, have been used extensively in systems biology to model a variety of biochemical networks, such as metabolic networks, gene regulatory networks and signal transduction networks. For many such networks, the available data quantity and quality is not sufficient to build detailed quantitative models such as systems of ordinary differential equations, which require many parameters that are frequently unknown. In addition, discrete models tend to be more intuitive and easily accessible to life scientists. There is ample evidence that such models can capture key dynamic features of biological networks and can be used successfully for hypothesis generation (G.Bhardwaj *et al*., submitted for publication; Saez-Rodriguez *et al*., 2009). Boolean networks and their generalization, the so-called multistate logical models (Thomas, 1973; Thomas and D'Ari, 1989), are the main types of finite dynamical systems that have been used successfully in modeling biological networks (e.g. Albert and Othmer, 2003; Barrett *et al*., 2005; Espinosa-Soto *et al*., 2004; Faure *et al*., 2006; Li *et al*., 2004; Mendoza *et al*., 1999; Saez-Rodriguez *et al*., 2009; Samal and Jain, 2008). Petri nets have also been shown to be a good modeling paradigm for this field (e.g. Chaouiya *et al*., 2008; Formanowicz *et al*., 2007; Grunwald *et al*., 2008; Kielbassa *et al*., 2009). Together, these model types represent a large class of discrete models in systems biology, which are capable of simulating deterministic as well as stochastic processes.

Several tools and techniques have been developed to simulate and analyze discrete models. For logical models, the open-source software GINsim (http://gin.univ-mrs.fr) is available, and for Petri nets, the user has access to a wide variety of software. The laboratory of M. Heiner provides software with a particular focus on applications to systems biology (http://www-dssz.informatik.tu-cottbus.de/). Analysis tools for logical models, including Boolean networks, are described in Naldi *et al*. (2007). In addition to a variety of simulation and visualization tools, other graph theoretic analysis tools are available for the identification of steady states and strongly connected components of the regulatory graph. There are algorithms to compute dead states (steady states), as well as T- and P-invariants, which can be computed via linear algebra methods. A survey of the use of Petri nets in systems biology is given by Peleg *et al*. (2005).

The purpose of this article is to describe a mathematical framework that encompasses both types of models and makes accessible a broad range of mathematical tools for model analysis, in order to complement existing tools in these domain areas. The fundamental observation underlying this framework is that logical models and k-bounded Petri nets are particular instantiations of what we shall call 'algebraic models,' i.e. time–discrete dynamical systems

$$f = (f_1, \ldots, f_n) : \mathbb{F}^n \longrightarrow \mathbb{F}^n$$

where each coordinate function f_i is a function of the n variables $x_1, \ldots, x_n$, each of which takes on values in a finite set $\mathbb{F}$ with algebraic structure, and each f_i is a polynomial. Aside from the mathematical simplicity of their definition, an important feature of polynomial dynamical systems is that one can employ a number of mathematical tools for their analysis. For our purposes, the principle tool is the capability to symbolically solve systems of nonlinear polynomial equations quite efficiently. This can be used, for instance, to compute the steady states and other features of an algebraic model.

We give algorithms that translate both logical models and Petri nets into this framework. Our algorithms are compatible with the algorithm in Chaouiya *et al*. (2008), which translates logical models into Petri nets. We furthermore provide several examples, using models from the literature, of how this translation can be used to analyze their dynamic properties.

The article is organized as follows. In Section 2, we briefly describe polynomial dynamical systems (PDSs). We then show how to use the algebraic framework in the analysis of logical models and

*To whom correspondence should be addressed.

k-bounded Petri nets in Sections 3 and 4, respectively; furthermore, we present concrete examples for each. We close with a discussion in Section 5.

2 METHOD

We consider models of biological systems, such as biochemical networks with n interacting molecular species, whose states can be described by an n-tuple with entries from a finite set $\mathbb{F}$. The model consists of a set of rules that allow the system to evolve from one state to the next, so that it can be represented as a time–discrete dynamical system $f:\mathbb{F}^n \longrightarrow \mathbb{F}^n$. Both logical models and k-bounded Petri nets are of this form. For example, in a model of a gene regulatory network, the set $\mathbb{F}$ is $\{0,1\}$, representing the state of a gene as either ON (1) or OFF (0).

If we consider f only as a set function, then there are few mathematical tools that can help to analyze f. One way to introduce mathematical structure, and thereby mathematical tools to study f, is by imposing the structure of a number system on $\mathbb{F}$, akin to the introduction of a coordinate system in affine space, which gives access to analytical methods in geometry. It can be shown that this can be done whenever the cardinality of $\mathbb{F}$ is a power of a prime number, and we will show in this article that it can always be accomplished in the cases of interest here (we will consider $|\mathbb{F}|$ to be a prime number to simplify the notations).

It is worth pointing out that this algebraic structure is used heavily in the case of Boolean networks. The evaluation of Boolean functions uses the fact that $\mathbb{F}=\{0,1\}$ is equipped with an addition, where 0 is the additive identity, and with a multiplication, where 1 is the multiplicative identity. The two are connected through the rule $1+1=0$.

Once $\mathbb{F}$ carries such a structure, we can make use of a fundamental property of finite fields: Let $h:\mathbb{F}^n\to\mathbb{F}$ be a function. Then, there exists a polynomial $g:\mathbb{F}^n\to\mathbb{F}$ such that $h(x)=g(x)$ for all $x\in\mathbb{F}^n$. One can find g using the formula $g(x)=\sum_{c\in\mathbb{F}^n} h(c)\prod_{j=1}^{n}(1-(x_j-c_j)^{p-1})$, where the right-hand side is computed modulo p. We will write h, $h(x)$ or $\sum_{c\in\mathbb{F}^n} h(c)\prod_{j=1}^{n}(1-(x_j-c_j)^{p-1})$, depending whether or not there is the need to specify x or the polynomial form of h.

Thus, we can represent f via its coordinate functions: $f=(f_1,\ldots,f_n):\mathbb{F}^n\longrightarrow\mathbb{F}^n$, where each f_i is a polynomial function, so that evaluation and analysis of f can be done using polynomial arithmetic over $\mathbb{F}$. We shall call f a *PDS* over $\mathbb{F}$ of dimension n. In the next sections, we will show that logical models and bounded Petri nets can be represented as PDSs. One can then use powerful symbolic computation software, from open source specialty packages such as *CoCoA* (http://cocoa.dima.unige.it/), *Singular* (http://www.singular.uni-kl.de) and *Macaulay2* (http://www.math.uiuc.edu/Macaulay2) to general commercial packages such as *Mathematica* and *Maple*. In this article, we will use *Macaulay2* for all computations.

3 ALGORITHM FOR LOGICAL MODELS

Logical models have been used for modeling biological phenomena and have been useful for obtaining valuable insight and qualitative information (Thomas and D'Ari, 1989). Such analysis is based on the topology of the network and the type of regulation of the different interactions in the network, which are believed to be the key features that determine dynamical properties (Albert and Othmer, 2003; Thomas and D'Ari, 1989). Logical models have the advantage of being intuitive and relatively simple to construct, even with no information about reaction rates or functionality, and still keeping qualitative information (Albert and Othmer, 2003). However, their analysis is not a trivial task. For example, the problem of finding the steady states of a logical model is NP-complete, that is, it is one of the hardest problems in the nondeterministic polynomial-time class (even to determine the existence of steady states is NP-complete; Zhao, 2005).

3.1 Definition of logical models

There are several ways to define logical models (Chaouiya *et al*., 2008; Comet *et al*., 2005; Naldi *et al*., 2007), and here we use the definition given in Naldi *et al*. (2007).

DEFINITION 3.1. *A logical model is a triple $(\mathcal{V},\mathcal{E},\mathcal{K})$, where:*

(1) *$\mathcal{V}=\{v_1,\ldots,v_n\}$ is the set of vertices or nodes. Each v_i has a maximum expression level, m_i. The set $\mathcal{S}=[0,m_1]\times\ldots\times[0,m_n]$ (Cartesian product) is called the* state space *of the logical model and its elements are called states.*

(2) *$\mathcal{E}$ is the set of arcs. The elements of $\mathcal{E}$ have the form (v_i,v_j,θ), where $1\le\theta\le m_i$ is called a threshold for (v_i,v_j). Let $\mathcal{I}(j)=\{v:(v,v_j,\theta)\in\mathcal{E}\}$, called input of v_j, be the set of vertices that have an edge ending in v_j. Notice that an arc (v_i,v_j) is allowed to have multiplicity corresponding to different thresholds; the number of thresholds is denoted by $m_{i,j}$. The thresholds are indexed in increasing order, $1\le\theta_{i,j,1}<\cdots<\theta_{i,j,m_{i,j}}\le m_i$; $\theta_{i,j,k}$ is the k-th threshold for the arc (v_i,v_j). By convention, we define $\theta_{i,j,m_{i,j}+1}=m_i+1$ (actually, $\theta_{i,j,m_{i,j}+1}$ can be defined as any number greater than m_i) and $\theta_{i,j,0}=0$. Let $x=(x_1,\ldots,x_n)$ be a state; we say that the k-th interaction for input v_i of v_j is active if $\theta_{i,j,k}\le x_i<\theta_{i,j,k+1}$; we denote this by $\Theta^{i,j}(x_i)=k$.*

(3) *$\mathcal{K}=\{K_i:\prod_{v_j\in\mathcal{I}(i)}[0,m_{j,i}]\to[0,m_i],i=1,\ldots n\}$ is the set of parameters.*

Consider a state $x=(x_1\ldots,x_n)$. The future value for node v_i, with $\mathcal{I}(i)=\{v_{i_1},\ldots,v_{i_r}\}$, is determined as follows: compute $\Theta^i(x_{i_1},\ldots,x_{i_r})=(\Theta^{i_1,i}(x_{i_1}),\ldots,\Theta^{i_r,i}(x_{i_r}))=(k_1,\ldots,k_r)$, indicating that the k_j-th interaction for input v_{i_j} of v_i is active; then compute $f_i(x)=K_i(k_1,\ldots,k_r)$.

This last value determines whether the value of v_i tends to increase, decrease or remain the same. To be precise, the future value of v_i is given by
$g_i(x)=\rho(x_i,f_i(x))$, where

$$\rho(t,u)=\begin{cases}t+1, & \text{if } u>t\\ t, & \text{if } u=t\\ t-1, & \text{if } u<t\end{cases}$$

With this notation, $g=(g_1,\ldots,g_n):\mathcal{S}\to\mathcal{S}$ is a finite dynamical system, and so is $f=(f_1,\ldots,f_n):\mathcal{S}\to\mathcal{S}$. Notice that g ensures that each variable changes at most one unit (continuity), whereas f does not.

The dynamics of a logical model is given by the *phase space*, defined as the graph $\mathcal{S}$ with an edge from x to y if $\{i:x_i\neq y_j\}=\{j\}$ for some j and $y_j=g_j(x)$ (asynchronous dynamics) or as the graph $\mathcal{S}$ with an edge from x to y if $y=g(x)$ (synchronous dynamics). Notice that g may be replaced by f if we do not require continuity.

An edge $i\to j$ is called *positive* (*negative*) if increasing the i-th entry of x causes $g_i(x)$ to increase (decrease) using the natural order on the integer sets $[0,m_i]$. If changing the i-th entry of x does not cause any change in $g_i(x)$, we say that the edge is nonfunctional.

3.2 Polynomial form of logical models

Let $(\mathcal{V},\mathcal{E},\mathcal{K})$ be a logical model as above. Choose a prime number p such that $p\ge m_i+1$ for all $1\le i\le n$ ($[0,m_i]$ has m_i+1 elements)

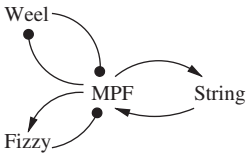

Fig. 1. A network for the *Drosophila* cell cycle (Chaouiya *et al*., 2004).

Table 1. Nonzero parameters for the *Drosophila* cell-cycle model

Target	Parameter
MPF	$K_M, K_{M,WS}, K_{M,S}, K_{M,W}$
Fizzy	$K_{F,M}$
Wee1	K_W
String	$K_{S,M}$

and let $\mathbb{F}=\mathbb{F}_p=\{0,1,\ldots,p-1\}$ be the field with p elements. Note that we may consider $\mathcal{S}\subseteq\mathbb{F}^n$.

Consider a vertex v_i and let g_i (or f_i if we do not require continuity) be its coordinate function. Our goal is to represent g_i as a polynomial in terms of its inputs, say $x_{i_1},\ldots,x_{i_r}$. That is, we need a polynomial function defined on $\mathbb{F}^r$ with values in $\mathbb{F}$. Denote $a\wedge b=\min\{a,b\}$, using the natural order on the set $\mathbb{F}$, viewed as integers. To extend the domain of g_i from $\prod_{v_j\in\mathcal{I}(i)}[0,m_{j,i}]$ to $\mathbb{F}^r$, we define $g_i(x_{i_1},\ldots,x_{i_r})=g_i(x_{i_1}\wedge m_{i_1},\ldots,x_{i_r}\wedge m_{i_r})$ for $(x_{i_1},\ldots,x_{i_r})\in\mathbb{F}^r$. The polynomial form of $g_i:\mathbb{F}^r\to\mathbb{F}$ is then

$$g_i(x)=\sum_{(c_{i_1},\ldots,c_{i_r})\in\mathbb{F}^r} g_i(c_{i_1},\ldots,c_{i_r})\prod_{v_j\in\mathcal{I}(i)}(1-(x_j-c_j)^{p-1}),$$

where the right-hand side is computed modulo p.

REMARK 3.2. *Many published logical models are Boolean, i.e. $m_i=1$ for all i. In this case, we can write $g_i=f_i$ nicely and explicitly in terms of the parameters*:

$$g_i(x)=\sum_{J\subseteq\mathcal{I}(i)} K_i(J)\prod_{v_j\in J}x_j\prod_{v_j\in\mathcal{I}(i)\setminus J}(1-x_j).$$

EXAMPLE 3.3. *Consider a Boolean logical model where x_2,x_3,x_4 regulate x_1, i.e. $\mathcal{I}(1)=\{2,3,4\}$. Suppose that $K_1(\{\})=K_1(\{3\})=K_1(\{4\})=K_1(\{3,4\})=1$ and $K_1(J)=0$ in all other cases. Using Remark* 3.2*, the polynomial form of g_1 is $g_1(x)=(1-x_2)(1-x_3)(1-x_4)+(1-x_2)x_3(1-x_4)+(1-x_2)(1-x_3)x_4+x_3(1-x_2)x_4$, which simplifies to $g_1=1+x_2$.*

3.3 Example

This example is based on a simplified model for the *Drosophila* cell cycle first presented in Chaouiya *et al*. (2004). The Boolean logical model is given in Figure 1 and Table 1.

Let $x_1=$MPF, $x_2=$Fizzy, $x_3=$Wee1, $x_4=$String. Notice that according to the network, g_1 depends on x_2, x_3 and x_4; i.e. $g_1(x)=g_1(x_2,x_3,x_4)$. The polynomial functions are: $g_1(x)=1+x_2$, $g_2(x)=x_1$, $g_3(x)=1+x_1$, and $g_4(x)=x_1$.

It follows that Wee1 $\to$ MPF and String $\to$ MPF are nonfunctional edges; i.e. changes in Wee1 or String do not affect MPF. Hence the network has only one functional (negative) circuit (see the Supplementary Material for a formal definition of functional circuit), rather than the three assumed in Chaouiya *et al*. (2004). In the transition from the complete model to the simplified model, some functionality may be lost and the algebraic framework can easily detect this. This is important because in order to deduce dynamical properties from the network structure it is necessary to have the correct wiring diagram.

Table 2. Nonzero and unknown parameters for the *Drosophila* cell cycle

Target	Parameter
MPF	$K_M, K_{M,S}$
Fizzy	$K_{F,M}$
Wee1	K_W
String	$K_{S,M}$
unknown parameters	$K_1=K_{M,W}, K_2=K_{M,WS}$ and $K_3=K_{M,FS}$

In Property 10 in Chaouiya *et al*. (2004), it is shown that the logical model has no steady states (this was done by proving that an equivalent Petri net is deadlock free). Algebraically, however, this corresponds to solving the system of equations

$$1+x_2=x_1, x_1=x_2, 1+x_1=x_3, x_1=x_4,$$

which has no solution, as the first two equations imply that $x_2=x_2+1$, showing that there are no steady states.

3.4 Algebra and parameters

An important feature of the algebraic framework is the ability to treat the parameters as variables and study all corresponding models at the same time, which we illustrate with a small logical model. (The same analysis can be done for Petri nets.)

Suppose, for example, that in the model for the *Drosophila* cell cycle (Section 3.3) some parameters are unknown.

The polynomial form of the logical model is: $g_1=1+x_2+x_3+K_1x_3+x_2x_3+K_1x_2x_3+K_3x_2x_4+K_1x_3x_4+K_2x_3x_4+K_1x_2x_3x_4+K_2x_2x_3x_4+K_3x_2x_3x_4$, $g_2=x_1$, $g_3=1+x_1$, and $g_4=x_1$. For the derivation of these equations, see the Supplementary Material.

Now, let us focus on finding steady states of this logical model. This model has three unknown parameters, and if we were to analyze each model, we would have to analyze $2^3=8$ logical models, as the complexity grows exponentially with respect to the number of unknown parameters. However, using the algebraic framework we can treat the parameters K_1,K_2,K_3 as variables and analyze the eight models at the same time.

Using polynomial algebra (see Supplementary Material), we find that an equivalent system is

$$K_1(K_3-1)=0,\ x_4(K_3-1)=0,\ K_1(x_4-1)=0,$$
$$x_3+x_4-1=0,\quad x_2+x_4=0,\quad x_1+x_4=0.$$

From this system, we can obtain valuable information. For example, if x is a steady state, then $x_1=x_2=x_4=x_3-1$; that is, the only possible steady states are $(0,0,1,0)$ and $(1,1,0,1)$. This is true no matter what the parameters K_1,K_2 and K_3 are.

We can also easily solve this system and see the steady states for any choice of parameters. For example, consider $K_1=1,K_3=0$. Since they do not satisfy the first equation, there are no steady states.

Table 3. Timing for the Gröbner basis computations

N	T_1	T_2	T_3	T_4	T_5
10	0.002	0.001	0.001	0.001	0.001
12	0.003	0.041	0.526	1.539	0.342
12	0.001	0.003	0.002	0.004	0.009
17	0.004	0.006	0.007	0.155	4.95
23	0.002	0.008	0.008	0.179	30.276
40	0.011	0.023	0.029	0.023	0.047

N = number of nodes, T_r = timings (s) for $f_1^r(x)-x_1=\cdots=f_n^r(x)-x_n=0$, where $r=1,\ldots,5$. The networks used were: fission yeast (Davidich and Bornholdt, 2007), budding yeast (Li *et al*., 2004), Th cell differentiation (Remy *et al*., 2006), Th cell differentiation (Mendoza, 2006), Th cell differentiation (Mendoza and Xenarios, 2006), T-cell receptor (Klamt *et al*., 2006), respectively. All networks were Boolean except the fourth one which had some nodes with three states.

Consider $K_1=K_2=K_3=0$; we have the system

$$x_4=0, x_3=1-x_4, x_2=x_4, x_1=x_4,$$

which has the unique steady state $x=(0,0,1,0)$.

We can also use the system above to determine for which choice of parameters a given state is a steady state of the system. For example, suppose we are interested in finding the parameters for which $(0,0,1,0)$ is a steady state, then we have the system $K_1(K_3-1)=0, 0(K_3-1)=0, K_1=0$; therefore, $K_1=0$ and the other parameters are 'free.' Hence, $(0,0,1,0)$ is a steady state for $K=(0,K_2,K_3)$.

Suppose now we are interested in finding the parameters for which $(1,1,0,1)$ is a steady state. In this case, we have the system $K_1(K_3-1)=0, K_3-1=0, K_1=0$; i.e. $K_1=0, K_3=1$ and K_2 is free. Hence, $(1,1,0,1)$ is a steady state for $K=(0,K_2,1)$ (note that in this case $(0,0,1,0)$ is a steady state as well).

3.5 Performance

We tested our algorithm to compute steady states of several published models, that is, we recorded the required time for computing a lexicographic Gröbner basis (which then allows backward substitution to solve the system of equations $f_1(x)-x_1=0$, …, $f_n(x)-x_n=0$.) The results are shown in Table 3. Furthermore, to compute limit cycles (using a parallel update) of length r, one can compute the lexicographic Gröbner basis for the system $f_1^t(x)-x_1=0$, …, $f_n^t(x)-x_n=0$, where $t=1,...,r$. It is clear that the solution set for $t=1$ is the set of steady states; the states in the solution set for $t=2$, which are not steady states, are periodic states with period two and so on.

Notice that in Table 3, the size of the network does not seem to be correlated to the computation time. This is due to the fact that polynomial algebra computations depend more strongly on the complexity of the polynomial equations and not as much on the size or complexity of the logical model. This means that the method can scale well to large networks and that it can complement other methods very well. (Typically, the more terms the polynomial form of the model has the more complex the Gröbner basis calculation. In the Boolean case, for instance, a conjunction leads to a polynomial with one term, whereas a disjunction results in three terms.)

4 ALGORITHM FOR K-BOUNDED PETRI NETS

4.1 Definition of Petri nets

A *k-bounded Petri net* is a 4-tuple (S,T,F,W), where:

(1) $S=\{s_1,\ldots,s_n\}$ is the set of places.
(2) $T=\{t_1,\ldots,t_m\}$ is the set of transitions.
(3) $F\subseteq(S\times T)\cup(T\times S)$ is the set of 'regular' arcs. $F^-\subseteq(S\times T)$ is the set of inhibitory arcs.
(4) $W:F\to\mathbb{N}$ gives the arc weights of the regular arcs. $W^-:F^-\to\mathbb{N}$ gives the arc weights of the inhibitory arcs.

For a transition t, define

$\bullet t=\{s\in S:(s,t)\in F\}$, $t\bullet=\{s\in S:(t,s)\in F\}$, $\bar{t}=\{s\in S:(s,t)\in F^-\}$.

The incidence matrix $A=(a_{ts})_{t\in T,s\in S}$ (or $A=(a_{ij})_{i,j=1}^{m,n}$) is an $m\times n$ matrix given by $a_{st}=W(t,s)-W(s,t)$ (we consider $W(e)=0$ when $e\notin F$). We denote by A_t the column of A^T (or the row of A) corresponding to a transition $t\in T$. Notice that $A_t=A^TU_t$, where the entry of U_t corresponding to t is 1 and all others are 0.

An assignment $x=(x_s)_{s\in S}\in\mathbb{N}^n$ [or $x=(x_1,\ldots,x_n)$] for all places is called a *marking*. Let x be a marking; a transition t is said to be *enabled* if $W(s,t)\le x_s$ for all $s\in\bullet t$ and $x_s\le W^-(s,t)-1$ for all $s\in\bar{t}$. Consider the function $C_t(x)$, defined by

$$C_t(x)=\begin{cases}1, \text{if } W(s,t)\le x_s \text{ for } s\in\bullet t \text{ and } x_s<W^-(s,t) \text{ for } s\in\bar{t},\\ 0, \text{otherwise.}\end{cases}$$

Thus, a transition t is enabled for x if $C_t(x)=1$.

The evolution of the Petri net is given by *firing* transitions; by firing an enabled transition t, we update the value of the places with regular arcs from/to t. The phase space of the Petri net is obtained by firing transitions in an asynchronous manner. Let x be a marking: if a transition t is enabled, then firing t results in the marking $x+A_t$. This implies that for any marking x and any transition t, the Petri net can always evolve from x to $f_t(x)=x+C_t(x)A_t=x+C_tA^TU_t$ (If $C_t(x)=0$, then $f_t(x)=x$; but if $C_t(x)=1$, then $f_t(x)=x+A_t=x+A^TU_t$). In the case where $x\neq y=f_t(x)$ we write $x\xrightarrow{t}y$. Denote by $f_{\mathcal{T}}$ the function obtained by composing $f_{t_1}\circ\cdots\circ f_{t_l}$, where $t_1,\ldots,t_l$ are the elements of $\mathcal{T}$. A trajectory is a path $x\xrightarrow{t_1}y\xrightarrow{t_2}\cdots$ and denoted by $x\xrightarrow{\mathcal{T}}$. For any trajectory $x\xrightarrow{\mathcal{T}}y$ we have $y=x+A^T\sum_{t\in\mathcal{T}}U_t$. The reachability graph for a marking x is the graph made up of all trajectories starting at x. The reachability graph of a set of markings S is the graph made up of all reachability graphs of all the elements of S.

We say that $(X_1,\ldots,X_m)>0$ if $X_i\ge0$ and at least for one i we have $X_i>0$. A P-invariant is an integer solution, $X>0$, of $AX=0$. A T-invariant is an integer solution, $Y>0$, of $A^TY=0$. A marking x is dead if no transition is enabled, that is, if $C_t(x)=0$ for all $t\in T$; this is equivalent to $f_t(x)=x$ for all $t\in T$.

A marking is *k-bounded* if the value of each place is at most k. The Petri net is *k-bounded* if any reachable marking (obtained by firing some sequence of transitions) from any k-bounded marking is also a k-bounded marking, that is, if $\{x:x_i\le k\}$ contains the vertices of its reachability graph, which is equivalent to $f_t([0,k]^n)\subseteq[0,k]^n$ for all transitions t. Examples of 1-bounded Petri nets are Boolean regulatory Petri nets, presented in Chaouiya *et al*. (2004); Remy *et al*. (2006); Steggles *et al*. (2007, 2006). When each place has its own 'k', we have a more general definition: for $K=(k_1,\ldots,k_n)$, a Petri net

is K-bounded if $f_t([0,k_1]\times\cdots\times[0,k_n])\subseteq[0,k_1]\times\cdots\times[0,k_n]$ for all transitions t. Examples of such Petri nets are multi-level regulatory Petri nets, presented in (Chaouiya *et al*., 2008; Comet *et al*., 2005). Our framework is applicable to K-bounded Petri nets (including k-bounded Petri nets).

For a given Petri net, the analysis of its dynamics, checking for dead markings and the type of liveness are some of the typical questions that can be viewed as algebraically as we will see next. Also, when modeling biological systems, there is not a unique marking that is of interest, but a whole family of markings corresponding to different initial states of the biological system. By looking at a Petri net as a polynomial dynamical system, we can study all those markings at the same time.

4.2 Polynomial form of K-bounded Petri nets

Since the algebraic framework relies on finite fields, we consider K-bounded Petri nets and focus on the analysis of markings in $\mathcal{S}=[0,k_1]\times\cdots\times[0,k_n]$. Let p be a prime number such that $k_i+1\leq p$ for all i and let $\mathbb{F}=\mathbb{F}_p$. Then, for all $t\in T$, $f_t:\mathcal{S}\subseteq\mathbb{F}^n\to\mathbb{F}^n$.

We need to extend the functions f_t to all of $\mathbb{F}^n$ and determine their polynomial form. It suffices to give algebraic structure to the function $C_t:\mathcal{S}\to[0,1]$ (since $f_t(x)=x+C_t(x)A_t$, the polynomial form of C_t will automatically give the polynomial form of f_t). To do this, consider the function $c_{[a,b]}(z)=1$ if $a\leq z\leq b$ and $c_{[a,b]}(z)=0$ otherwise (i.e. $c_{[a,b]}$ is the characteristic function of $[a,b]$). Since $C_t(x)=\prod_{s\in{}^\bullet t}c_{[W(s,t),k_s]}(x_s)\prod_{s\in\bar{t}}c_{[0,W(s,t)-1]}(x_s)$, we only need to give algebraic structure to $c_{[a,b]}$ (the polynomial form of C_t will be given by the product of the polynomial forms of its factors). It is not difficult to see that the polynomial function $\sum_{a\leq r\leq b}(1-(z-r)^{p-1})$ is equal to 1 if $a\leq z\leq b$ and 0 otherwise. Hence, the polynomial form of $c_{[a,b]}$ is $c_{[a,b]}(z)=\sum_{a\leq r\leq b}(1-(z-r)^{p-1})$. This gives the polynomial form of $C_t(x)$, which, in turn, gives the polynomial form of $f_t(x)=x+C_t(x)A_t$, where $f_t:\mathbb{F}^n\to\mathbb{F}^n$.

EXAMPLE 4.1. *Consider a transition t of a* 1*-bounded Petri net such that* ${}^\bullet t=\{x_1,x_2\}$ *and* $\bar{t}=\{x_3,x_4\}$ *(with weights equal to* 1*). Then* $C_t(x)=c_{[1,1]}(x_1)c_{[1,1]}(x_2)c_{[0,0]}(x_3)c_{[0,0]}(x_4)$. *Since* $c_{[0,0]}(z)=1-z$ *and* $c_{[1,1]}(z)=z$, *it follows that* $C_t(x)=x_1x_2(1-x_3)(1-x_4)$. *Hence the polynomial form of* f_t *is given by* $f_t(x)=x+x_1x_2(1-x_3)(1-x_4)A_t$.

Now, we can state and solve algebraically some Petri net problems. For example, the problem of finding dead markings becomes the problem of solving polynomial equations, which can be easily addressed within the algebraic framework, as was the case for logical models. More precisely, we have the following remark.

REMARK 4.2. *A marking x of a K-bounded Petri net is dead if and only if $x\in\mathcal{S}$ and $f_t(x)=x$ for all $t\in T$. Let $\varphi(x)=\prod_{i=1}^n c_{[0,k_i]}(x_i)$. Note that $\varphi(x)=1$ if $x\in\mathcal{S}$ and $\varphi(x)=0$ otherwise. Then the set of dead markings is given by the solutions of the system of polynomial equations*

$$f_{t_1}(x)=x\ ,\quad \ldots\ ,\quad f_{t_m}(x)=x\ ,\quad \varphi(x)=1.$$

REMARK 4.3. *If the polynomial form of f_t for all transitions t is known, then we can easily recover A using the equation $C_t(x)A_t=f_t(x)-x$ and hence find the P- and T-invariants.*

4.3 Example

Consider the Petri net model in Chaouiya *et al*. (2008) for the core lambda switch

$S=\{CI,Cro\}$.
$T=\{t_1=t^+_{CI},\ t_2=t^-_{CI,\{(CI,1),(Cro,1)\}},\ t_3=t^+_{Cro},\ t_4=t^-_{Cro,\{(Cro,2)\}},\ t_5=t^-_{Cro,\{(CI,1)\}},\ t_6=t^-_{Cro,\{(CI,1),(Cro,2)\}}\}$.
$F=\{(t_1,CI),\ (Cro,t_1),\ (CI,t_2),\ (Cro,t_2),\ (t_2,Cro),\ (t_3,Cro),\ (Cro,t_4),\ (t_4,Cro),\ (CI,t_5),\ (Cro,t_5),\ (CI,t_6),\ (Cro,t_6),\ (t_6,Cro)\}$.
$F^-=\{(CI,t_1),(CI,t_3),(Cro,t_3),(CI,t_4),(Cro,t_5)\}$.

$W(e)=2$ for $e\in\{(Cro,t_4),(Cro,t_6)\}$ and $W(e)=1$ for the other arcs in F. Also, $W^-(e)=2$ for $e\in\{(Cro,t_3),(Cro,t_5)\}$ and $W(e)=1$ for the other arcs in F^-.

It follows that A^T is given by

$$\begin{array}{c} \\ CI \\ Cro \end{array}\!\!\begin{array}{c} \begin{array}{cccccc} t_1 & t_2 & t_3 & t_4 & t_5 & t_6 \end{array} \\ \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & -1 & -1 \end{pmatrix} \end{array}.$$

This Petri net is (1,2)-bounded (Chaouiya *et al*., 2008) and, therefore, we can use the algebraic framework. Notice that $\varphi(x)=c_{[0,1]}(x_1)c_{[0,2]}(x_2)=(x_1+1)^2$. Let us compute the polynomial form of f_{t_5}. First, notice that $c_{[0,0]}(z)=(1-z)(1+z)$, $c_{[1,1]}(z)=z(2-z)$, $c_{[2,2]}(z)=z(1-z)$, $c_{[1,2]}(z)=z^2$ and $c_{[0,1]}(z)=(1+z)^2$. Then, $C_{t_5}(x)=c_{[1,1]}(x_1)c_{[1,2]}(x_2)c_{[0,1]}(x_2)$ which reduces to $C_{t_5}(x)=x_1(2-x_1)x_2(2-x_2)$. Since $A_{t_5}=(0,-1)$, it follows that $f_{t_5}(x)=x+x_1(2-x_1)x_2(2-x_2)(0,-1)=(x_1,x_2-x_1(2-x_1)x_2(2-x_2))$. The polynomial forms for all the transitions are

$$\begin{aligned}
f_1(x) &= (x_1+(1-x_1)(1+x_1)(1-x_2)(1+x_2),x_2),\\
f_2(x) &= (x_1-x_1(2-x_1)x_2^2,x_2),\\
f_3(x) &= (x_1,x_2+(1-x_1)(1+x_1)(1+x_2)^2),\\
f_4(x) &= (x_1,x_2-(1-x_1)(1+x_1)x_2(1-x_2)),\\
f_5(x) &= (x_1,x_2-x_1(2-x_1)x_2(2-x_2)),\\
f_6(x) &= (x_1,x_2-x_1(2-x_1)x_2(1-x_2)).
\end{aligned}$$

To find the dead markings, we have to solve the system

$$\begin{aligned}
(1-x_1)(1+x_1)(1-x_2)(1+x_2) &= 0,\\
x_1(2-x_1)x_2^2 &= 0,\\
(1-x_1)(1+x_1)(1+x_2)^2 &= 0,\\
(1-x_1)(1+x_1)x_2(1-x_2) &= 0,\\
x_1(2-x_1)x_2(2-x_2) &= 0,\\
x_1(2-x_1)x_2(1-x_2) &= 0,\\
(x_1+1)^2-1 &= 0.
\end{aligned}$$

The last equation corresponds to the requirement that $x\in\mathcal{S}$. Using *Macaulay2*, we obtain the marking $(1,0)$ as the only dead marking.

5 DISCUSSION

The problem of giving mathematical structure to logical models has been studied by several authors (e.g. Egri-Nagy and Nehaniv, 2008). For the purpose of computation and analysis, the structure proposed here provides a class of simple and easily defined mathematical objects that can model both logical models and bounded Petri nets. It has the advantage that it makes accessible the theoretical concepts, algorithms and software from polynomial algebra, such as Gröbner

basis theory, which underlies many of the algorithms for solving systems of polynomial equations.

It is important to mention that in this article we focus on finding steady states and dead markings, not because that is the only application of our framework, but because it can be translated directly into the algebraic problem of solving polynomial equations without much difficulty; the framework we propose can be also used to answer other questions. For instance, theory, algorithms and software were used in Laubenbacher and Stigler (2004) and subsequent papers to give a solution to the problem of reverse-engineering of gene regulatory networks from experimental time course data. This allows us to find all logical models that satisfy certain properties as done by SMBioNet (http://smbionet.lami.univ-evry.fr) with the additional advantage of allowing a systematic study and classification of such models. One can also solve the reverse problem, that is, given a family of models, by using our framework, we find properties that they all satisfy (see Section 3.4). Furthermore, the theory inherent in the algebraic framework can give rise to theorems. For example, the algebraic structure was used to give an exact formula for the structure of the phase space of linear systems (Elspas, 1959; Hernández-Toledo, 2005) and lower and upper bounds for the number of limit cycles of conjunctive and disjunctive Boolean networks (Jarrah *et al*., 2010). Also, the family of nested canalizing functions has been shown to have an algebraic structure (toric variety) that allows their characterization (Jarrah *et al*., 2007). These functions appear frequently in Boolean models of regulatory networks and their dynamics have desirable properties. Functionality of circuits can also be studied within this framework (see Supplementary Material).

Polynomial algebra can, therefore, complement the existing analysis tools for logical models and Petri nets, and we have shown several examples of its use. It is also worth mentioning that Petri nets are a special case of bipartite models, consisting of two sets of nodes, representing 'places' and 'events,' respectively, connected by directed edges. More general bipartite models are quite common in systems biology (e.g. Warren *et al*., 2009), and they might be amenable to analysis with similar methods. It is also worth mentioning that, while the examples used here are all gene regulatory networks, there are examples of logical/Petri net models of other kinds of molecular networks (e.g. Simao *et al*., 2005), so that the analysis framework described here is more widely applicable.

A potential disadvantage of the algebraic framework is that it is less intuitive than either logical models or Petri nets. Even in the Boolean case it is often difficult to give a biological interpretation to a Boolean function in polynomial form, even if the equivalent Boolean expression is quite meaningful. However, with appropriate software, the typical user wanting to analyze a logical model or Petri net does not need to explicitly manipulate polynomial functions or even be aware that they are used in the analysis. On the other hand, some questions or aspects of logical models and Petri nets may not currently have a direct counterpart in the algebraic framework; for example, the asynchronous update commonly used in logical models does not currently have a direct counterpart in the polynomial algebra framework. This deserves further investigation.

ACKNOWLEDGEMENTS

The authors are grateful to the anonymous referees for many insightful comments and suggestions.

Conflict of Interest: none declared.

REFERENCES

Albert,R. and Othmer,H. (2003) The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in *Drosophila melanogaster*. *J. Theor. Bio.*, **223**,1–18.

Barrett,C. *et al*. (2005) The global transcriptional regulatory network for metabolism in *Escherichia coli* exhibits few dominant functional states. *Proc. Natl Acad. Sci. USA*, **102**,19103–19108.

Chaouiya,C. *et al*. (2004) Qualitative modelling of genetic networks: from logical regulatory graphs to standard Petri nets. In *ICATPN'04*, Springer, Berlin, Germany, pp. 137–156.

Chaouiya,C. *et al*. (2008) Petri net modelling of biological regulatory networks. *J. Discrete Algorithms*, **6**, 165–177.

Comet,J.-P. *et al*. (2005) Modeling multi-valued genetic regulatory networks using high-level Petri nets. In Ciardo,G. and Darondeau,P. (eds) *Proceedings of the International Conference on the Application and Theory of Petri Nets*, Springer, Berlin, Germany, pp. 208–227.

Davidich,M. and Bornholdt,S. (2007) Boolean network model predicts cell cycle sequence of fission yeast. *PLoS One*, **3**, e1672.

Elspas,B. (1959) The theory of autonomous linear sequential networks. *IRE Trans. Circuit Theory*, **CT-6**, 45–60.

Egri-Nagy,A. and Nehaniv,C. (2008) Algebraic properties of automata associated to Petri nets and applications to computation in biological systems. *Biosystems*, **94**, 297–307.

Espinosa-Soto,C. *et al*. (2004) A gene regulatory network model for cell-fate determination during *Arabidopsis thaliana* flower development that is robust and recovers experimental gene expression profiles. *Plant Cell*, **16**, 2923–2939.

Faure,A. *et al*. (2006) Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics*, **22**, 124–131.

Formanowicz,D. *et al*. (2007) Petri net based model of the body iron homeostasis. *J. Biomed. Inform.*, **40**, 476–485.

Grunwald,S. *et al*. (2008) Petri net modelling of gene regulation of the duchenne muscular dystrophy. *Biosystems*, **92**, 189–205.

Hernández-Toledo,A. (2005) Linear finite dynamical systems. *Communications in Algebra*, **33**, 2977–2989.

Jarrah,A. *et al*. (2010) The Dynamics of Conjunctive and Disjunctive Boolean Network Models. *Bull. Math. Bio.*, [Epub ahead of print, doi: 10.1007/s11538-010-9501-z, January 20, 2010].

Jarrah,A. *et al*. (2007) Nested canalyzing, unate cascade, and polynomial functions. *Physica D*, **233**, 167–174.

Kielbassa,J. *et al*. (2009) Modeling the U1 snRNP assembly pathway in alternative splicing in human cells using Petri nets. *Comput. Biol. Chem.*, **33**, 46–61.

Klamt,S. *et al*. (2006) A methodology for the structural and functional analysis of signaling and regulatory networks. *BMC Bioinformatics*, **7**, 56.

Laubenbacher,R. and Stigler,B. (2004) A computational algebra approach to the reverse-engineering of gene regulatory networks. *J. Theor. Biol.*, **229**, 523–537.

Li,F. *et al*. (2004) The yeast cell-cycle network is robustly designed. *Proc. Natl Acad. Sci. USA*, **101**, 4781–4786.

Mendoza,L. (2006) A network model for the control of the differentiation process in Th cells. *Biosystems*, **84**, 101–114.

Mendoza,L. and Xenarios,I. (2006) A method for the generation of standardized qualitative dynamical systems of regulatory networks. *Theor. Biol. Med. Model.*, **3**, Article no. 13.

Mendoza,L. *et al*. (1999) Genetic control of flower morphogenesis in arabidopsis thaliana: a logical analysis. *Bioinformatics*, **15**, 593–606.

Naldi,A. *et al*. (2007) Decision diagrams for the representation and analysis of logical models of genetic networks. In*CMSB'07*, Vol. 4695 of *LNCS/LNBI*, Springer, Berlin, Germany, pp. 233–247.

Peleg,M. *et al*. (2005) Using Petri net tools to study properties and dynamics of biological systems. *J. Am. Med. Informatics Assoc.*, **12**, 181–199.

Remy,E. *et al*. (2006) From logical regulatory graphs to standard Petri nets: dynamical roles and functionality of feedback circuits. In *Transactions on Computation Systems Biology VII (TCSB)*, Springer, Berlin, Germany, pp. 55–72.

Saez-Rodriguez,J. *et al*. (2009) Discrete logic modelling as a means to link protein signalling networks with functional analysis of mammalian signal transduction. *Mol. Syst. Biol.*, **5**, 331.

Samal,A. and Jain,S. (2008) The regulatory network of *E. coli* metabolism as a Boolean dynamical system exhibits both homeostasis and flexibility of response. *BMC Syst. Biol.*, **2**, Article 21.

Simao,E. *et al*. (2005) Qualitative modelling of regulated metabolic pathways: application to the tryptophan biosynthesis in *E. coli. Bioinformatics*, **21**, 190–196.

Steggles,L. *et al*. (2007) Qualitatively modelling and analysing genetic regulatory networks: a Petri net approach. *Bioinformatics*, **23**, 336–343.

Steggles,L. *et al*. (2006) Modelling and analysing genetic networks: From Boolean networks to Petri nets. In *Computational Methods in Systems Biology. International Conference*, Springer, Berlin, Germany, pp. 127–141.

Thomas,R. (1973) Boolean formalisation of genetic control circuits. *J. Theor. Biol.*, **42**, 565–583.

Thomas,R. and D'Ari,R. (1989) *Biological Feedback*. CRC Press, Boca Raton, FL.

Warren,P. *et al*. (2009) Flux networks in metabolic graphs. *Phys. Biol.*, **6**, 46006.

Zhao,Q. (2005) A remark on "Scalar Equations for Synchronous Boolean Networks with Biological Applications" by C. Farrow, J. Heidel, J. Maloney, and J. Rogers. *IEEE Trans. Neural Netw.*, **16**, 1715–1716.