Logic Programming for Boolean Networks

Katsumi Inoue
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan
ki@nii.ac.jp

Abstract

The Boolean network is a mathematical model of biological systems, and has attracted much attention as a qualitative tool for analyzing the regulatory system. The stable states and dynamics of Boolean networks are characterized by their attractors, whose properties have been analyzed computationally, yet not much work has been done from the viewpoint of logical inference systems. In this paper, we show direct translations of Boolean networks into logic programs, and propose new methods to compute their trajectories and attractors based on inference on such logic programs. In particular, point attractors of both synchronous and asynchronous Boolean networks are characterized as supported models of logic programs so that SAT techniques can be applied to compute them. Investigation of these relationships suggests us to view Boolean networks as logic programs and vice versa.

1 Introduction

Analysis of static and dynamic behavior of systems has become more and more important recently in various domains including *systems biology*. The *Boolean network* (BN) was proposed as a mathematical model of genetic networks and complex adaptive systems [Kauffman, 1969; 1993], and has been used as a discrete model of gene regulatory, signal transduction and protein interaction networks. The BN offers a simple yet powerful tool based on Boolean logic, and has a network structure consisting of nodes that correspond to genes or proteins. Each node in a BN takes a value of 1 or 0, meaning that the gene is or is not expressed. The value of a node at a time step is determined according to a regulation rule that is a Boolean function of the values of its input nodes at the previous time step. Computational properties and physical behaviors of the BN have been well studied in several fields including biology, physics, and bioinformatics, e.g., [Kauffman, 1993; Harvey and Bossomaier, 1997; Shmulevich *et al.*, 2002; Gershenson, 2002; Irons, 2006; Akutsu *et al.*, 2007; Garg *et al.*, 2008; Remy and Ruet, 2008].

The stable states and dynamics of BNs are characterized by their *attractors*, which play an essential role in biological systems. The number and length of attractors of a random BN have been analyzed computationally [Kauffman, 1993; Irons, 2006], yet not much work has been done from the viewpoint of logical inference systems. Recently, detection of attractors has been investigated in bioinformatics, and several studies connect attractor computation with various techniques based on advances of Propositional Satisfiability (SAT) and Answer Set Programming (ASP). These works can be classified into two types: those simulating dynamic behaviors of systems as trajectories of a series of inference [Fayruzov *et al.*, 2009; Dubrova and Teslenko, 2010] or those computing attractors with the shortest length (i.e., *point attractors*) [Tamura and Akutsu, 2009; Melkman *et al.*, 2010]. Those algorithms can be used for computing attractors of *synchronous BNs*, which are the simplest model of BNs.

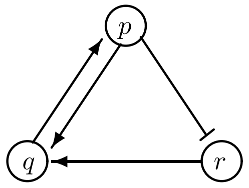
In this paper, we formalize BNs as logic programs, and give a novel characterization of their attractors based on the semantics of logic programs. In contrast to the previous approaches that translate individual BN-related problems into equivalent theories, we will provide a direct encoding of BNs into logic programs and give a *semantical characterization* of BNs so that any BN-related problem can be analyzed based on this semantics. Then, several theoretical results and computational tools of logic programming can easily be transferred to BNs. In particular, point attractors of both *synchronous* and *asynchronous* Boolean networks are naturally characterized as *supported models* of logic programs [Apt *et al.*, 1988], enabling us to apply SAT techniques to compute them. Thus, this work opens an integrated way to apply AI techniques to various computational problems of BNs.

On the other hand, from the viewpoint of logic programming, this work not only shows an important application of logic programming to systems biology and dynamic domains, but opens a new research direction toward a *dynamic semantics* of logic programs. In fact, most existing semantics regard a program as a static specification or an axiom set of a problem, and do not regard each rule as a transition function of a dynamic system. Considering the dynamic semantics in this work, physical behaviors that have been studied for BNs can be transferred to analyze dynamics of logic programs.

In the rest of this paper, Section 2 reviews the BN. Section 3 considers synchronous BNs and characterizes their attractors in terms of the semantics of corresponding logic programs. Section 4 represents asynchronous BNs and their attractors. Sections 5 and 6 discuss related and future work.

2 Boolean Networks

A *Boolean network* (BN) is a pair $N = (V, F)$, where $V = \{v_1, \ldots, v_n\}$ is a finite set of nodes (or *genes*) and $F = \{f_1, \ldots, f_n\}$ is a corresponding set of Boolean functions (called *regulation functions*). Let v_i^t represent the state of v_i at time t, which takes the value of either 1 (expressed) or 0 (not expressed). A vector (or *state*) $\mathbf{v}(t) = (v_1^t, \ldots, v_n^t)$ is the overall expression level of all nodes in N at time step t. Note that there are 2^n possible states for each time step. The state of a node v_i at the next time step $t+1$ is determined by $v_i^{t+1} = f_i(v_{i_1}^t, \ldots, v_{i_k}^t)$, where $v_{i_1}, \ldots, v_{i_k}$ are the set of *input nodes* of v_i that are the genes directly connected to (or regulating) v_i, and the number k is called the *indegree* of v_i. When the indegree of v_i is 0, no update is done at any time step. A BN is usually represented by a graph with two types of edges, which are positive and negative, in which $v_1 \longrightarrow v_2$ means that v_1^t positively takes part in the regulation function for v_2^{t+1} and $v_1 \dashv v_2$ means that v_1^t negatively takes part in the regulation function for v_2^{t+1}. An example of a BN is $N_1 = (V_1, F_1)$, where $V_1 = \{p, q, r\}$ and F_1 is as follows.



$$p^{t+1} = q^t,$$
$$q^{t+1} = p^t \wedge r^t,$$
$$r^{t+1} = \neg p^t. \qquad (1)$$

The state of each gene can be updated either synchronously or asynchronously. In a *synchronous BN* (SBN), the states of all genes are updated simultaneously, while in an *asynchronous BN* (ABN) not all nodes are necessarily updated at a time. Each update of nodes in a BN N changes its state to another. A consecutive sequence of states obtained by state transitions is called a *trajectory* of N. State transition in an SBN is *deterministic*, and a trajectory starting from any state is uniquely determined. On the other hand, it is *nondeterministic* in an ABN. Let $w \in \{0,1\}^n$ be a state, and $R(w)$ be the states reachable in all trajectories starting from w. Then, a set of states S is an *attractor* if $R(w) = S$ holds for every $w \in S$ [Garg *et al.*, 2008]. If any trajectory from a node in an attractor S composes a single loop, $w_0, \ldots, w_{p-1}, w_p (= w_0)$, where $p = |S|$ $(1 \le p \le 2^n)$, S is called a *point attractor* (or *singleton attractor*) when $p = 1$, and is called a *cycle attractor* when $p > 1$. The set of states that reach the same attractor is called its *basin of attraction* [Kauffman, 1993]. For example, when the BN N_1 is synchronous, starting from the initial state $\mathbf{v}(0) = (0,1,1)$, the trajectory becomes $(0,1,1)$, $(1,0,1)$, $(0,1,0)$, $(1,0,1)$, …, and $(1,0,1) \to (0,1,0) \to (1,0,1)$ is a cycle attractor (Figure 1 below). N_1 has another, point attractor $(0,0,1)$ (Figure 1 above) whose basin of attraction is $\{(1,1,1), (1,1,0), (1,0,0), (0,0,0), (0,0,1)\}$.

Note that any state in an SBN deterministically belongs to the basin of attraction of only one attractor, which is either a cycle attractor or a point attractor. Several interpretations of attractors have been reported in the literature [Kauffman, 1993; Shmulevich *et al.*, 2002]; for example, each attractor represents a cell type, a type of memory, or a cellular state such as proliferation, apoptosis and differentiation.

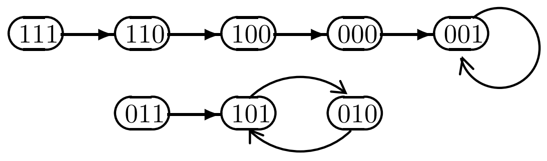


Figure 1: State transition diagram for the BN N_1

3 Synchnorous BNs

In this section, we characterize synchronous BNs based on the semantics of logic programming. Our concern is to express regulation functions as rules of logic programs. To this end, we examine the appropriateness of existing semantics for logic programs.

3.1 Normal Logic Programs

We consider a first-order language and denote the Herbrand base (the set of all ground atoms) as $\mathcal{A}$. A *(normal) logic program* (NLP) is a set of rules of the form

$$A \leftarrow A_1 \wedge \cdots \wedge A_m \wedge \neg A_{m+1} \wedge \cdots \wedge \neg A_n \qquad (2)$$

where A and A_i's are atoms $(n \ge m \ge 0)$. For any rule R of the form (2), the atom A is called the *head* of R and is denoted as $h(R)$, and the conjunction to the right of $\leftarrow$ is called the *body* of R and we represent the positive and negative literals in the body as $b^+(R) = \{A_1, \ldots, A_m\}$ and $b^-(R) = \{A_{m+1}, \ldots, A_n\}$, respectively. Let $ground(P)$ be the set of ground instances of all rules in an NLP P. An (*Herbrand*) *interpretation* I is a subset of $\mathcal{A}$, and is called an (*Herbrand*) *model* of P if I satisfies all ground rules from P, that is, for any rule $R \in ground(P)$, $b^+(R) \subseteq I$ and $b^-(R) \cap I = \emptyset$ imply $h(R) \in I$.

One of the most important criteria that any model theoretic semantics of logic programming should satisfy is the "supportedness". An Herbrand model $I \subseteq \mathcal{A}$ of an NLP P is a *supported model* [Apt *et al.*, 1988] of P if for any ground atom $A \in I$, there exists a rule $R \in ground(P)$ such that (a) $h(R) = A$, (b) $b^+(R) \subseteq I$, and (c) $b^-(R) \cap I = \emptyset$. On the other hand, an Herbrand model I is a *stable model* [Gelfond and Lifschitz, 1988] of P if I is the least model of the program $P^I = \{(h(R) \leftarrow \bigwedge_{B \in b^+(R)} B) \mid R \in ground(P),\ b^-(R) \cap I = \emptyset\}$. It is known that every stable model is a supported model [Marek and Subrahmanian, 1992]. For example, $\{\, p \leftarrow p,\ q \leftarrow \neg p \,\}$, has the supported models $\{p\}$ and $\{q\}$, but only the latter is its stable model.

The following operational semantics of NLPs has been given in [Apt *et al.*, 1988]. Given an NLP P and an interpretation I, the *immediate consequence operator* (or T_P *operator*) $T_P : 2^{\mathcal{A}} \to 2^{\mathcal{A}}$ is defined as $T_P(I) = \{\, h(R) \mid R \in ground(P),\ b^+(R) \subseteq I,\ b^-(R) \cap I = \emptyset\}$. Then, I is a model of P iff I is a pre-fixed point of T_P, i.e., $T_P(I) \subseteq I$. By definition, I is supported iff $I \subseteq T_P(I)$. Hence, I is a supported model of P iff I is a fixpoint of T_P, i.e., $T_P(I) = I$. Moreover, I is a model of $Comp(P)$, which is the Clark's *completion* of P, iff $T_P(I) = I$. This means that the supported models of P are precisely the models of $Comp(P)$.

We next show that the T_P operator is useful to characterize attractors of synchronous BNs. As a related concept, the sequence of applications of the T_P operator is considered by Blair *et al.* [1997]: For an Herbrand interpretation I, the *orbit* of I with respect to P is the sequence $\langle {T_P}^k(I)\rangle_{k\in\omega}$ where ${T_P}^0(I) = I$ and ${T_P}^{k+1}(I) = T_P({T_P}^k(I))$ for $k = 0, 1, \ldots$.

3.2 Characterizing Trajectories and Attractors

Our idea here is to express the regulation functions of a BN as rules of an NLP. To this end, we need to transform a Boolean function f_i for a node v_i in disjunctive normal form (DNF). Since this is always possible, in the following we assume that any regulation function is a DNF formula. Suppose a synchronous BN $N = (V, F)$. For each node $v_i \in V$, suppose its regulation function $f_i \in F$ is given as a DNF formula

$$v_i^{t+1} = \bigvee_{j=1}^{l_i} B_{i,j}^t, \quad B_{i,j}^t = \bigwedge_{k=1}^{m_j} v_{i,j,k}^t \ \wedge \bigwedge_{k=m_j+1}^{n_j} \neg v_{i,j,k}^t, \quad (3)$$

where $v_{i,j,k} \in V$ and $n_j \geq m_j \geq 0$ for $j = 1, \ldots, l_i$. Note that j can be 0 for a node v_i, that is, v_i has no input nodes. In this case, v_i is called a *constant node*, and the indegree of a constant node is 0. Since there is no regulation function for a constant node, it stays in a constant state, i.e., always expresses or never expresses. Let $V_C \subseteq V$ be the constant nodes of N. Notice that V_C is fixed and is smaller than V.

Now, for each $f_i \in F$, let $\rho_T(f_i)$ be the rule set defined as either of the following sets:

$$\begin{array}{ll} \{\, (v_i(s(t)) \leftarrow B_{i,j}(t)) \mid 1 \leq j \leq l_i \,\}, & \text{if } v_i \in (V \setminus V_C); \\ \{\, (v_i(s(t)) \leftarrow v_i(t)) \,\}, & \text{if } v_i \in V_C, \end{array}$$

where $s(t)$ is the successor of t, i.e., $t+1$, and each $v_i \in V$ is now used as a predicate and $B_{i,j}(t)$ is the formula obtained by replacing every $v_{i,j,k}^t$ appearing in $B_{i,j}^t$ with $v_{i,j,k}(t)$. Then the NLP for N is given as $P_T(N) = \bigcup_{v_i\in V} \rho_T(f_i)$.

Let $V(t) = \{v_i(t) \mid v_i \in V\}$. Given a state $\mathbf{v}(t) = (v_1^t, \ldots, v_n^t)$ at time step t, the interpretation $I_{\mathbf{v}(t)} \subseteq V(t)$ is defined as $I_{\mathbf{v}(t)} = \{v_i(t) \in V(t) \mid v_i^t = 1\}$. Then,

Proposition 3.1 $I_{\mathbf{v}(s(t))} = T_{P_T(N)}(I_{\mathbf{v}(t)}) \cap V(s(t))$.

For example, for the BN N_1 (1), the NLP $P_T(N_1)$ is

$$p(s(t)) \leftarrow q(t), \ \ q(s(t)) \leftarrow p(t) \wedge r(t), \ \ r(s(t)) \leftarrow \neg p(t).$$

Now, given the initial state $\mathbf{v}(0) = (0, 1, 1)$, the orbit of $I_{\mathbf{v}(0)}$ with respect to $P_T(N_1)$ is $\{q(0), r(0)\}$, $\{p(s(0))\} \cup \{r(s^k(0)) \mid k \geq 1\}$, $\{q(s(s(0)))\} \cup \{r(s^k(0)) \mid (k = 1) \vee (k \geq 3)\}$, $\{p(s(s(s(0))))\} \cup \{r(s^k(0)) \mid k \geq 1\}, \ldots$. Then, the sequence $\langle I_{\mathbf{v}(s^k(0))}\rangle_{k\in\omega}$ is $\{q(0), r(0)\}$, $\{p(s(0)), r(s(0))\}$, $\{q(s(s(0)))\}$, $\{p(s(s(s(0)))), r(s(s(s(0))))\}$, $\ldots$, which simulates the trajectory of N_1 from $\mathbf{v}(0)$.

Since each $I_{\mathbf{v}(t)}$ only contains atoms that are true at time step t, it turns out that the time argument can be omitted from each rule in $\rho_T(f_i)$. For each $f_i \in F$ given as (3), let $\rho(f_i)$ be the set of *propositional* rules obtained by deleting the time arguments (t) and $s(t)$ from all literals appearing in $\rho_T(f_i)$, and define $P(N) = \bigcup_{v_i\in V} \rho(f_i)$. Also, for any state $\mathbf{v}(t)$, put $I^t = \{v_i \in V \mid v_i^t = 1\}$.

Theorem 3.2 *Let N be a synchronous BN, and $\mathbf{v}(t)$ any state. Then, the orbit of I^t with respect to $P(N)$ is precisely the trajectory of N starting from $\mathbf{v}(t)$.*

Theorem 3.2 implies that every attractor can be obtained as an attracting cycle in an orbit of some initial state $\mathbf{v}(0)$, and in particular that a point attractor is given as a fixed point in such an orbit. For example, for N_1, the propositional NLP $P(N_1)$ is now defined as

$$p \leftarrow q, \quad q \leftarrow p \wedge r, \quad r \leftarrow \neg p. \quad (4)$$

Then, given the initial state $\mathbf{v}(0) = (0, 1, 1)$, the orbit of I^0 with respect to $P(N_1)$ is $\{q, r\}$, $\{p, r\}$, $\{q\}$, $\{p, r\}$, $\ldots$, which is exactly the trajectory of N_1 from $\mathbf{v}(0)$, indicating that the repeat $\{p, r\} \rightarrow \{q\} \rightarrow \{p, r\}$ corresponds to the cycle attractor of N_1. Similarly, starting from $(1, 1, 1)$, the orbit of $\{p, q, r\}$ reaches the fixed point $\{r\}$, which corresponds to the point attractor $(0, 0, 1)$ of N_1.

3.3 Computing Point Attractors

To detect all attractors of a BN based on Theorem 3.2, we need to prepare $I_{\mathbf{v}(0)}$ for each initial state $\mathbf{v}(0)$, but there are 2^n such possible initial states. Here, we show that detection of all point attractors of a BN can be characterized without specifying the set of initial states.

In the following, we will identify a state $\mathbf{v}(t)$ with an interpretation I^t at a point attractor, and denote it as I whenever the time step is not important.

Theorem 3.3 *Let $N = (V, F)$ be a synchronous BN. Then, I is a point attractor of N iff I is a supported model of $P(N)$.*

The proof of Theorem 3.3 can be established as follows: (i) By Theorem 3.2 and the definition of point attractors, I is a point attractor of N iff $I = T_{P(N)}(I)$ holds; (ii) $I = T_{P(N)}(I)$ iff I is a supported model of $P(N)$ (see Section 3.1). Theorem 3.3 precisely characterizes the set of all point attractors of a BN in terms of one propositional program. As mentioned earlier, all supported models of an NLP $P(N)$ can be enumerated as the models of its completion $Comp(P(N))$, which is given as follows.

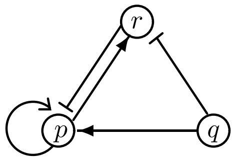
$$\begin{aligned} &Comp(P(N)) \\ &\equiv \bigwedge_{v_i\in(V\setminus V_C)} \left(v_i \leftrightarrow \bigvee_{j=1}^{l_i} B_{i,j} \right) \wedge \bigwedge_{v_i\in V_C} (v_i \leftrightarrow v_i) \\ &\equiv \bigwedge_{v_i\in(V\setminus V_C)} \left(v_i \leftrightarrow \bigvee_{j=1}^{l_i} B_{i,j} \right). \qquad (5) \end{aligned}$$

Then, using a SAT solver, computation of point attractors can be automated by computing the models of the formula (5)[1] without testing for each initial state one by one. In $Comp(P(N))$, each completed formula for a non-constant node v_i represents the logical equivalence between the state of v_i and its previous state. On the other hand, the additional rule $(v_i \leftarrow v_i)$ in $\rho(f_i)$ for each constant node $v_i \in V_C$ preserves the truth value of v_i in an initial state $\mathbf{v}(0)$ through the orbit of $I_{\mathbf{v}}(0)$. This rule becomes the formula $(v_i \leftrightarrow v_i)$ in $Comp(P(N))$, and can be equivalently removed from (5).

[1]The same translation from a synchronous BN into SAT has been directly suggested in [Tamura and Akutsu, 2009] without getting through the semantics of logic programming.

The rule $(v_i \leftarrow v_i)$ for each $v_i \in V_C$ thus prevents the derivation of $\neg v_i$ in $Comp(P(N))$, and is actually indispensable in obtaining the supported models as the next example shows.

Example 3.1 Consider the next BN $N_2 = (\{p, q, r\}, F_2)$:



$$p^{t+1} = p^t \vee q^t \vee \neg r^t,$$
$$r^{t+1} = p^t \wedge \neg q^t. \qquad (6)$$

In N_2, q is a constant node. Let P_2 be the set of rules:

$$p \leftarrow p, \quad p \leftarrow q, \quad p \leftarrow \neg r, \quad r \leftarrow p \wedge \neg q. \qquad (7)$$

The supported model of P_2 is $\{p, r\}$, which corresponds to the point attractor $(1, 0, 1)$. However, another point attractor $(1, 1, 0)$ cannot be obtained from P_2, since $\{p, q\}$ is not a supported model of P_2. We see that $\{p, q\}$ is a supported model of $P(N_2) = P_2 \cup \{ q \leftarrow q \}$.

In Example 3.1, $P(N_2)$ has no stable model. In fact, any tautological rule can be ignored in the stable model semantics. It is known that, I is a stable model of an NLP P iff I is a supported model of both P and some $P' \subseteq P$ such that P' does not have any loop in its positive dependency graph[2] [Lin and Zhao, 2004]. For example, in $P(N_1)$ (4), the rules $Q = \{ p \leftarrow q, \ \ q \leftarrow p \wedge r \}$ have a loop in the positive dependency graph. Then, $P(N_1)' = P(N_1) \setminus Q = \{ r \leftarrow \neg p \}$ has the unique supported model $\{r\}$, which is the stable model of $P(N_1)$. For P_2 (7), $\{p, r\}$ is not a supported model of $P_2' = P_2 \setminus \{p \leftarrow p\}$, and is not a stable model of P_2. We now define a similar property for BNs: an attractor is called *non-self-dependent* if any its expressing node does not depend on itself in the positive dependency graph. We characterize this by the stable model semantics for NLPs with *choice rules*[3] [Simon *et al.*, 2002]. Intuitively, the choice rule $(0\{p\}1 \leftarrow \ )$ represents that p is or is not contained in each stable model.

Theorem 3.4 *Let $N = (V, F)$ be a synchronous BN. Then, I is a non-self-dependent point attractor of N iff I is a stable model of* $P(N) \cup \{(0\{v\}1 \leftarrow \ ) \mid v \in V_C\}$.[4]

For Example 3.1, $\{p, r\}$ is neither a stable model of $P(N_2)$ nor a stable model of $P(N_2) \cup \{q\}$, and hence is not a non-self-dependent point attractor of N_2. On the other hand, the stable model $\{p, q\}$ of $P(N_2) \cup \{q\}$ is non-self-dependent.

3.4 From NLPs to BNs

We now show a converse translation from NLPs to BNs. For this, we assume that the Herbrand base $\mathcal{A}$ is finite, and its elements can be numbered as $A_1, \ldots, A_n$. Then, any interpretation $I \subseteq \mathcal{A}$ can be identified with the n-tuple $([A_1]_I, \ldots, [A_n]_I)$, where $[A_i]_I$ is the truth value of A_i in I, i.e., $[A_i]_I = 1$ iff $A_i \in I$ and $[A_i]_I = 0$ iff $A_i \notin I$. Given a propositional NLP P, construct the synchronous BN $N(P) = (\mathcal{A}, F(P))$, where $F(P)$ is defined as follows. For each $A \in \mathcal{A}$, suppose the set of rules in P whose heads are A is given by $\{(A \leftarrow B_1), \ldots, (A \leftarrow B_k)\}$, where each B_j $(1 \le j \le k)$ is a conjunction of literals. Then, the regulation function for A is defined as $A^{t+1} = (B_1^t \vee \cdots \vee B_k^t)$ when $k \ge 1$, and is assigned 0 when $k = 0$ (A is set as a 0-node).

Theorem 3.5 *Let P be a propositional NLP. Then, I is a supported model of P iff I is a point attractor of $N(P)$.*

An obvious merit of representation of BNs in terms of logic programming semantics (and vice versa) is that we can apply rich resources of NLPs to BNs (and vice versa). Such theories include complexity results, computational procedures, and several relations between two programs such as equivalence and generality. Here, we just apply a well-known complexity result for NLPs to BNs. Detection of a point attractor in a BN is known to be NP-hard [Milano and Roli, 1999], and this holds even for a BN with the maximum indegree 2 [Tamura and Akutsu, 2009]. To the best of our knowledge, however, the next results have never been stated in the literature, yet the problems are important to know whether some specific genes can be expressed or not in genetic networks.

Theorem 3.6 *Let $N = (V, F)$ be a synchronous BN, and $v \in V$ any node in N. Deciding if v is expressed in some point attractor (resp. all point attractors) of N is NP-complete (resp. co-NP-complete).*

Theorem 3.6 is proved based on the complexity results for the supported model semantics [Marek and Truszczyński, 1991], and the (co-)NP-completeness also holds for the same inference problems of non-self-dependent attractors.

4 Asynchronous BNs

It has been argued in biology that *asynchronous BNs* (ABNs) are closer to real biological phenomena than synchronous BNs (SBNs), but due to their complexities to model and analyze, less studies have been done for ABNs compared with SBNs. In ABNs, different genes take different time to make the state transition from one expression level to another, and hence delay always occurs. To model asynchronicity, several *update schemes* have been proposed [Gershenson, 2002; Garg *et al.*, 2008], and different updating schemes may lead to different attractors. Here, we assume that update of a state is done by nondeterministically choosing nodes to be updated at each time step. This update scheme is called *generalized asynchronous* in [Gershenson, 2002], and contains many other update schemes as its special cases.

Remember that any attractor of an SBN is either a point attractor or a cycle attractor. By contrast, in an ABN, there are attractors (called *loose attractors* [Harvey and Bossomaier, 1997] or *complex loops* [Garg *et al.*, 2008]) which are neither cycle attractors nor point attractors.

Example 4.1 Consider the BN $N_3 = (\{p, q\}, F_3)$ in Figure 2 (left). The state transition diagram of N_3 with the synchronous update is depicted as the thick lines in Figure 2 (right), while the asynchronous update has additional transitions written in the thin lines as well as self loops corresponding null updates (not shown in the figure). The label

[2]The directed graph consisting of the positive edges in a BN N is exactly the positive dependency graph of $P(N)$.

[3]A choice rule is an instance of cardinality constraints [Simon *et al.*, 2002], which is explained in Section 4.

[4]Each rule of the form $(v_i \leftarrow v_i)$ for any $v_i \in V_C$ can be omitted from $P(N)$ in the presence of choice rules in Theorem 3.4.

F_3: $p^{t+1} = \neg q^t$,
$q^{t+1} = \neg p^t$.

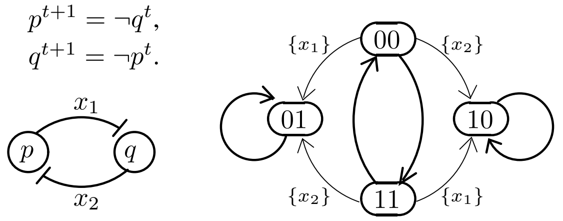


Figure 2: Synchronous and asynchronous transitions for N_3

of an arc in a thin line represents the set of selected rules in that update. Any transition in a thick line uses the whole set $\{x_1, x_2\}$. Note that the next state of $(0, 1)$ or $(1, 0)$ is only itself whichever set of rules is chosen for update.

The two point attractors, $(0, 1)$ and $(1, 0)$, of N_3 in the synchronous update are also point attractors in the asynchronous update. The cycle attractor $(0, 0) \to (1, 1) \to (0, 0)$ in the synchronous update is not an attractor in the asynchronous update since there are transitions outgoing from this cycle. In fact, the states $(0, 0)$ and $(1, 1)$ are soon trapped by one of the point attractors by a sequential update.

We now show a translation of ABNs into logic programs. A *cardinality constraint* [Simon *et al.*, 2002] is an expression of the form $l\{L_1, \ldots, L_n\}u$, where L_i $(1 \le i \le n)$ is a literal and l and u are integers such that $0 \le l \le u \le n$. Intuitively, $l\{L_1, \ldots, L_n\}u$ is satisfied by a set I of atoms if the number of the literals satisfied by I in $\{L_1, \ldots, L_n\}$ is between the lower bound l and the upper bound u. Cardinality constraints are used in heads and bodies of rules as a basic construct in answer set programming (ASP).

Suppose an ABN $N = (V, F)$ where each $f_i \in F$ is given by (3). For each $v_i \in V$, we introduce a new predicate x_i intending that $x_i^t = 1$ represents that v_i is chosen for update with f_i at time step t. For N, we define the program $P_T^{nd}(N)$ as the union of the rules (8) and (9) defined for each $v_i \in (V \setminus V_C)$, the rules (10) defined for each constant node $c_i \in V_C$, and the rule (11):

$$\begin{aligned} v_i(s(t)) &\leftarrow x_i(t) \wedge B_{i,j}(t) \quad \text{for } j = 1, \ldots, l_i, && (8)\\ v_i(s(t)) &\leftarrow \neg x_i(t) \wedge v_i(t), && (9)\\ c_i(s(t)) &\leftarrow c_i(t), && (10)\\ l\,X(t)\,u &\leftarrow\ , && (11) \end{aligned}$$

where $X(t) = \{x_i(t) \mid v_i \in (V \setminus V_C)\}$ and $0 \le l \le u \le n'$ for $n' = |V \setminus V_C|$.

Here, the Boolean function f_i is used to update $v_i(t)$ if $x_i(t)$ is true by (8); otherwise $(\neg x_i(t))$, the value of v_i is not updated by (9). The value of a constant node does not change by (10). In (11), the cardinality constraint $l\,X(t)\,u$ represents that between l and u nodes can be updated at each time step, given two parameters l and u. When $l = 0$, null updates are allowed, i.e., updating no node at a time step. When $l = 1$, at least one non-constant node is chosen for update at each time step. In particular, $l = u = 1$ represents a *sequential update* [Harvey and Bossomaier, 1997; Garg *et al.*, 2008], in which only one node is chosen for each update. When $l = u = n'$, all non-constant nodes are updated simultaneously, hence an SBN is expressed as a special case of ABNs.

For the program $P_T^{nd}(N)$, the T_P operator for NLPs cannot be directly applied since a rule with a cardinality constraint in its head has a nondeterministic effect.[5] In our case, however, the cardinality constraint appears only at the rule (11), and then can be replaced with choice rules and integrity constraints. In the following, to simplify the discussion, we fix the parameters in (11) as $l = 0$ and $u = n'$, that is, any number of nodes can be chosen for update at each time step. Then, $(0\,X(t)\,n' \leftarrow\ )$ (11) can be replaced with

$$0\{x_i(t)\}1 \ \leftarrow\ , \quad \text{for each } v_i \in (V \setminus V_C). \qquad (12)$$

Under the supported model semantics, the rules (12) can be further translated into

$$x_i(t) \leftarrow x_i(t) \quad \text{for each } v_i \in (V \setminus V_C). \qquad (13)$$

Let $P_T^d(N)$ be the union of the rules (8), (9), (10), and (13). Note that $P_T^d(N)$ is an NLP. Now, given any state $\mathbf{v}(t) = (v_1^t, \ldots, v_n^t)$, suppose $I_{\mathbf{x}(t)} \subseteq X(t)$ are selected at time step t. Then, the next state $I_{\mathbf{v}(s(t))}$ is uniquely determined as:

Proposition 4.1 $I_{\mathbf{v}(s(t))} = T_{P_T^d(N)}(I_{\mathbf{v}(t)} \cup I_{\mathbf{x}(t)}) \cap V(s(t))$.

Since each $I_{\mathbf{v}(t)}$ only mentions a state at time step t, we can drop the time argument from each literal of the form $v_i(t)$ or $c_i(t)$ in the rules (8), (9) and (10) of $P_T^d(N)$ as in the case of SBNs. Moreover, dropping the time argument is also applied to the selection literal $x_i(t)$'s in the rules (8), (9), and (13). Let $P^A(N)$ be the resulting propositional program.

Unlike Theorem 3.2 for SBNs, the propositional NLP $P^A(N)$ cannot precisely simulate the trajectories of an ABN N. However, point attractors of N can be identified by the deterministic immediate consequence operator with $P^A(N)$:

Proposition 4.2 *Let $N = (V, F)$ be an ABN. Then, I is a point attractor of N iff $I = T_{P^A(N)}(I \cup Y) \cap V$ for any set $Y \subseteq \{x_i \mid v_i \in V \setminus V_C\}$.*

Given a BN $N = (V, F)$, we can interpret N as either a synchronous BN or an asynchronous BN. We denote it as N^s if N is synchronous, and as N^a if asynchronous. Now, we show the relationship between them as follows.

Proposition 4.3 *I is a point attractor of N^a iff I is a point attractor of N^s.*

Proposition 4.3 can be proved by the fact that, for each $v_i \in V$, either $(v_i \leftarrow x_i \wedge B_{i,j})$ for $j = 1, \ldots, l_i$ or $(v_i \leftarrow \neg x_i \wedge v_i)$ is selected. In the former case, the state of v_i is not changed in N^s because I is a point attractor of N^a. In the latter case, the state of v_i is not updated by $(v_i \leftarrow v_i)$. Then I must be a point attractor of N^s. The converse holds by Proposition 4.2.

Although a fact similar to Proposition 4.3 has already been stated in [Gershenson, 2002; Garg *et al.*, 2008], we formally obtain the result by the semantics of BNs in this paper. Hence, computation of point attractors of an ABN N can be done by Theorem 3.3 via the translation $P(N)$ for the SBN. This fact also justifies a reason why computation of point attractors is more important in SBNs than that of cycle attractors.

[5]For a program P with cardinality constraints, we could use the *nondeterministic consequence operator* $T_P^{nd} : 2^{\mathcal{A}} \to 2^{2^{\mathcal{A}}}$ in [Marek *et al.*, 2007] and define orbits as branching trees induced by T_P^{nd}.

5 Discussion

Computing Attractors and Stable States. Several methods have recently been proposed to compute attractors of synchronous BNs based on ASP and SAT techniques. In [Dubrova and Teslenko, 2010], attractors are searched on state transition diagrams of BNs with bounded model checking, in which the length of trajectories is incrementally varied. This computation process is repeated until an upper bound, and then each obtained trajectory is checked whether an attractor is included or not. In contrast, our Theorem 3.3 shows that only one propositional formula $Comp(P(N))$ is necessary to get all point attractors, but cycle attractors cannot be obtained in this way.[6] Particular focuses are put on point attractors in [Tamura and Akutsu, 2009; Melkman *et al.*, 2010] based on more elaborate translations of BNs into SAT. Conversely, SAT is translated into BNs in [Milano and Roli, 1999; Tamura and Akutsu, 2009]. Fayruzov *et al.* [2009] use ASP for computing attractors, but their conflict resolution is different from the original BNs, and hence it is not clear how complex regulation functions can be expressed in their approach. ASP is also used to analyze biological networks that are not BNs: Gebser *et al.* [2008] detect inconsistencies in biological networks, and Ray *et al.* [2011] compute supported and stable models from reaction networks that respectively correspond to states with and without self-supporting loops.

In principle, each previous work translates a *particular BN-related problem* into an equivalent computational problem at the *meta-level*. For example, [Fayruzov *et al.*, 2009] translates $a^{t+1} = b^t$ into $activates(a, b, t)$ and defines several axioms for updates, e.g., $(act(Y, T + 1) \leftarrow act(X, T) \wedge activates(X, Y, T) \wedge \neg conflict(Y, T) \wedge \neg mos\_act\_th(Y))$. In contrast, we simply map the regulation functions directly to $(a \leftarrow b)$, and then give the *semantics* of them. This work thus enables us to analyze *any BN-related problem* on the semantics of logic programs, and provide an integrated way to apply theories and techniques developed in logic programming to various computational problems in BNs.

Control of BNs. Akutsu *et al.* [2007] have defined the problem to find a control strategy of a BN, which leads to the given goal state from the given initial state. Some *external control nodes* are also given and their values are manipulated in [Akutsu *et al.*, 2007] to achieve the goal state, which is somewhat similar to AI planning. Viewing a BN as an NLP, a related problem can be defined by modifying the decision problem in Theorem 3.6 to finding of initial states of constant nodes that realize expression of a *target node* v. This can be achieved by finding a supported model containing v.

Simulation and Equivalence of BNs. Blair *et al.* [1997] did a pioneer work to view logic programs as *cellular automata* (CAs), which are special cases of synchronous BNs considered in this paper, and pointed out the importance of orbits in capturing inferential behaviors of NLPs. However, their main concern is not on computational properties of attractors, but is on simulation of an orbit for an NLP by an orbit for some Horn program. As far as attractors are concerned, we can define equivalence between two BNs with control nodes, and can capture it by *relativized equivalence* under the supported model semantics [Truszczyński and Woltran, 2008].

Positive and Negative Loops. It has been observed that (I) presence of positive loops in a BN is linked to *multistability*, i.e., existence of multiple point attractors involved in differentiation, and that (II) presence of negative loops is the source of *periodic oscillations*, i.e., existence of cycle attractors involved in homeostasis [Remy and Ruet, 2008]. Here, the sign of a loop is defined as the product of the signs of its edges. On the other hand, [You and Yuan, 1994] have shown that (I') existence of even loops in an NLP implies multiple stable models, and that (II') existence of odd loops implies presence of undefined literals. In (I'), however, even loops in NLPs are assumed to be *non-zero*, i.e., containing at least two negative edges. As the supported models of NLPs correspond to point attractors of BNs, by considering positive loops without negative edges, the property (I) of BNs can be transferred to NLPs, providing a necessary condition to have multiple supported models. In fact, supported models and stable models are not much different from each other, with only difference on the treatment of such positive loops [Lin and Zhao, 2004].

Relating the properties (II) of BNs and (II') of NLPs, cycle attractors in BNs are closely related to undefined literals in *regular models* [You and Yuan, 1994] as well as the *stable class* semantics [Baral and Subrahmanian, 1992]. For example, $P_4 = \{p \leftarrow \neg p\}$ has no supported model and the BN $N(P_4)$ has no point attractor by Theorem 3.5, but the orbit of $I_0 = \emptyset$ or $I_1 = \{p\}$ with respect to P_4 becomes I_0, I_1, I_0, $I_1, \ldots$, and the cycle attractor $(0) \rightarrow (1) \rightarrow (0)$ is found in it. Hence, viewing an NLP as a BN, we get more information from the program, and can get a meaning even when it is inconsistent under the supported or stable model semantics.

6 Conclusion

We have seen that there are so many common properties between BNs and NLPs. This comparison can be promoted by considering the similarity between the fixed points of BNs and the T_P operator of NLPs. We have revived the notion of orbits in this paper, but have also shown that orbits themselves are not necessary to characterize point attractors of BNs. In fact, supported models of logic programs are useful for that, and this work opens an important application of the supported model semantics in which non-stable supported models play an important role. With the "*attractor semantics*", logic programs that are inconsistent under the supported or stable model semantics can have meanings as they have cycle attractors. Such semantics had ever been investigated in some work on logic programming [Baral and Subrahmanian, 1992; You and Yuan, 1994], but have not been focused on recently.

An obvious merit of NLPs compared with BNs is that first-order representation is possible in NLPs. Then, we could consider *first-order BNs* to describe dynamics of systems with infinite domains or to have compact representation for common update patterns. In the research of BNs, *probabilistic BNs* [Shmulevich *et al.*, 2002] have been investigated, in which multiple Boolean functions can be assigned to each node and

[6]We have tested our SAT-based approach to biological data in [Dubrova and Teslenko, 2010], and obtained better performance to get point attractors, e.g., all 7 point attractors for Drosophila melanogaster (52 nodes) are obtained within 0.05 sec by MiniSat2.

one function is selected randomly each time. Alternatively, to express such nondeterminism in a non-stochastic way, we could consider *disjunctive BNs* to allow indefinite effects. Relating these extensions of BNs with probabilistic or disjunctive logic programming is also an interesting future topic. Finally, *abductive* and *inductive* logic programming could be applied to abduction and induction on BNs to infer missing nodes and missing regulation functions from observations and training examples, respectively.

Acknowledgments

This research is supported in part by the 2008-2011 JSPS Grant-in-Aid for Scientific Research (A) No. 20240016. The author would like to thank Tatsuya Akutsu, Chiaki Sakama and Takehide Soh for their useful discussions, and anonymous reviewers for their helpful comments.

References

[Akutsu *et al.*, 2007] Akutsu, T., Hayashida, M., Ching, W-K. and Ng, M. K., Control of Boolean networks: Hardness results and algorithms for tree structured networks, *Journal of Theoretical Biology*, 244:670–679, 2007.

[Apt *et al.*, 1988] Apt, K. R., Blair, H. A. and Walker, A., Towards a theory of declarative knowledge, in: J. Minker (ed.), *Foundations of Deductive Databases and Logic Programming*, Morgan Kaufmann, pp.89–148, 1988.

[Baral and Subrahmanian, 1992] Baral, C. and Subrahmanian, V. S., Stable and extension class theory for logic programs and default logics, *Journal of Automated Reasoning*, 8:345–366, 1992.

[Blair *et al.*, 1997] Blair, H. A., Dushin, F. and Humenn, P. R., Simulations between programs as cellular automata, *Proceedings of LPNMR '97*, LNAI 1265, pp.115–131, 1997.

[Dubrova and Teslenko, 2010] Dubrova, E. and Teslenko, M., A SAT-based algorithm for finding attractors in synchronous Boolean networks, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2010.

[Fayruzov *et al.*, 2009] Fayruzov, T., De Cock, M., Cornelis, C. and Vermeir, D., Modeling protein interaction networks with answer set programming, *Proceedings of 2009 IEEE International Conference on Bioinformatics and Biomedicine*, pp.99–104, 2009.

[Garg *et al.*, 2008] Garg, A., Di Cara, A., Xenarios, I., Mendoza, L. and De Micheli, G., Synchronous versus asynchronous modeling of gene regulatory networks, *Bioinformatics*, 24(17):1917–1925, 2008.

[Gebser *et al.*, 2008] Gebser, M., Schaub, T., Thiele, S., Usadel, B. and Veber, P., Detecting inconsistencies in large biological networks with answer set programming, *Proceedings of ICLP '08*, LNCS 5366, pp.130–144, 2008.

[Gelfond and Lifschitz, 1988] Gelfond, M. and Lifschitz, V., The stable model semantics for logic programming, *Proceedings of ICLP '88*, pp.1070–1080, MIT Press, 1988.

[Gershenson, 2002] Gershenson, C., Classification of random Boolean networks, in: *Artificial Life VIII*, pp.1–8, MIT Press, 2002.

[Harvey and Bossomaier, 1997] Harvey, I. and Bossomaier, T., Time out of joint: Attractors in asynchronous random Boolean networks, in: *Proceedings of the 4th European Conference on Artificial Life*, pp.67–75, MIT Press, 1997.

[Irons, 2006] Irons, D. J., Improving the efficiency of attractor cycle identification in Boolean networks, *Physica D*, 217:7–21, 2006.

[Kauffman, 1969] Kauffman, S. A., Metabolic stability and epigenesis in randomly constructed genetic nets, *Journal of Theoretical Biology*, 22(3):437–467, 1969.

[Kauffman, 1993] Kauffman, S. A., *The Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, 1993.

[Lin and Zhao, 2004] Lin, F. and Zhao, Y., ASSAT: computing answer sets of a logic program by SAT solvers, *Artificial Intelligence*, 157:115–137, 2004.

[Marek *et al.*, 2007] Marek, V. W., Niemelä, I. and Truszczyński, M., Logic programs with monotone abstract constraint atoms, *Theory and Practice of Logic Programming*, 8(2):167–199, 2007.

[Marek and Subrahmanian, 1992] Marek, W. and Subrahmanian, V. S., The relationship between stable, supported, default and autoepistemic semantics for general logic programs, *Theor. Comp. Sci.*, 103(2):365–386, 1992.

[Marek and Truszczyński, 1991] Marek, W. and Truszczyński, M., Computing intersection of autoepistemic expansions, *Proceedings of LPNMR '91*, pp.37–50, 1991.

[Melkman *et al.*, 2010] Melkman, A. A., Tamura, T. and Akutsu, T., Determining a singleton attractor of an AND/OR Boolean network in $O(1.587^n)$ time, *Information Processing Letters*, 110(14-15):565–569, 2010.

[Milano and Roli, 1999] Milano, M. and Roli, A., Solving the satisfiability problem through Boolean networks, *Proceedings of AI*IA '99*, pp.72–83, LNAI 1792, 1999.

[Ray *et al.*, 2011] Ray, O., Soh, T. and Inoue, K., Analyzing pathways using ASP-based approaches, in: *Proceedings of the International Conference on Algebraic and Numeric Biology 2010*, LNCS, to appear, Springer, 2011.

[Remy and Ruet, 2008] Remy, E. and Ruet, P., From elementary signed circuits to the dynamics of Boolean regulatory networks, *Bioinformatics*, 24:220–226, 2008.

[Shmulevich *et al.*, 2002] Shmulevich, I., Dougherty, E. R. and Zhang, W., From Boolean to probabilistic Boolean networks as models of genetic regulatory networks, *Proceedings of the IEEE*, 90(11):1778–1792, 2002.

[Simon *et al.*, 2002] Simons, P., Niemelä, I. and Soininen, T., Extending and implementing the stable model semantics, *Artificial Intelligence*, 138(1–2):181–234, 2002.

[Tamura and Akutsu, 2009] Tamura, T. and Akutsu, T., Detecting a singleton attractor in a Boolean network utilizing SAT algorithms, *IEICE Trans.*, 92-A(s):493–501, 2009.

[Truszczyński and Woltran, 2008] Truszczyński, M. and Woltran, S., Hyperequivalence of logic programs with respect to supported models, *Annals of Mathematics and Artificial Intelligence*, 53:331–365, 2008.

[You and Yuan, 1994] You, J. H. and Yuan, L., A three-valued semantics for deductive database and logic programs, *J. Comput. Syst. Sci.*, 49:334–361, 1994.