AND-NOT logic framework for steady state analysis of Boolean network models

Alan Veliz-Cuba1,2,∗, Kristina Buschur2, Rose Hamershock2, Ariel Kniss2, Esther Wolff2, Reinhard Laubenbacher2

1University of Nebraska-Lincoln, 203 Avery Hall, Lincoln, NE 68588, USA.
2Virginia Bioinformatics Institute, Washington Street, Blacksburg, VA 24061, USA.

Received: 15 Nov. 2012, Revised: 13 Jan. 2013, Accepted: 2 Mar. 2013
Published online: 1 Jul. 2013

Abstract: In this paper we propose the class of AND-NOT networks for modeling biological systems and show that it provides several advantages. Some of the advantages include: Any finite dynamical system can be written as an AND-NOT network with similar dynamical properties. There is a one-to-one correspondence between AND-NOT networks, their wiring diagrams, and their dynamics. Results about AND-NOT networks can be stated at the wiring diagram level without losing any information. Results about AND-NOT networks are applicable to any Boolean network. We apply our results to a Boolean model of Th-cell differentiation.

Keywords: Steady states, AND-NOT logic, NAND gate, Boolean model

1 Introduction

Discrete models have a long and successful history in systems biology, beginning with Boolean network representations of molecular networks [1] and their later generalization, so-called logical models [2]. They are qualitative, time-discrete models that focus attention on the qualitative features of the system, such as the wiring diagram, and are particularly suitable for the analysis of steady state behavior of molecular networks. However, as models become larger it is increasingly difficult to analyze them. In order to keep the analysis of such networks tractable, many studies have focused on specific classes of networks such as: single-switch, unate, nested canalizing, threshold, AND, AND-OR, and linear networks [3,4,5,6,7,8,9,10,11,12,13,14]. In order to be useful for modeling, a family of networks has to be “sufficiently general” for modeling biological interactions and “simple enough” for theoretical analysis. In this paper we propose the family of AND-NOT networks as such family. AND-NOT networks are a particular class of Boolean networks that are constructed using only the AND (∧) and NOT (¬) operators.

A biological justification for the use of AND-NOT networks is that there is evidence that for genes that are regulated by more than one other gene, the different binding sites exhibit synergistic effects between the different regulators [15,16,17]. This fact motivated the study of conjunctive Boolean networks, that is, networks whose logical rules are constructed using exclusively the AND operator [5], where explicit formulas for steady states are given; also, upper and lower bounds for the number and length of limit cycles are provided. But conjunctive Boolean networks cannot account for inhibitory regulation and the resulting negative feedback loops, which are common in gene regulatory networks. Allowing the NOT operator, in addition to the AND operator (i.e. using AND-NOT networks), can make the family of networks sufficiently general to be useful for modeling [18].

For a formal argument that the family of AND-NOT networks is general enough for modeling, we will show that any discrete model (finite dynamical system, to be precise) can be represented by an AND-NOT network. More precisely, we present an algorithm that assigns to a given general discrete model an AND-NOT network which has the same number of steady states, together with an algorithmic correspondence between steady states of the two networks. This is achieved by adding nodes to the network as needed. The potential drawback of this algorithm is of course that the network size can potentially get significantly larger, thereby potentially negating any computational advantage gained by the specialized logic. However, since molecular networks

∗ Corresponding author e-mail: aveliz-cuba2@unl.edu
have typically small in-degree, this growth in the number of network nodes to be added is modest in the case of molecular network models. We demonstrate this through an analysis of several published models and random networks.

To argue that AND-NOT networks are simple enough for theoretical analysis, we will show how using the specialized logic of AND-NOT networks can provide better theoretical results. For example, in [19], it was shown that an upper bound for the number of steady states can easily be computed for AND-NOT networks (which is not true for arbitrary networks). Also, in [20], it was shown that the exact number of steady states of AND-NOT networks are encoded in the topological features of the wiring diagram, and that, in some cases, the problem of finding the exact number of steady states can be transformed to the problem of finding maximal independent sets of the wiring diagram, which has been extensively studied [21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. In this paper we will show how the specialized logic of AND-NOT networks can give us better upper bounds for the number of steady states; more precisely, we provide an upper bound for AND-NOT networks that improves on previous upper bounds. Furthermore, we show how this upper bound for AND-NOT networks can actually be used for general networks. We use our results to analyze a Boolean model of Th-cell differentiation. Another theoretical advantage of AND-NOT networks is that they are in a one-to-one correspondence with their wiring diagrams. This observation has several implications, one of which is the possibility to relate dynamic network properties with features of the wiring diagram [5, 20]. Also, from a given signed wiring diagram one can unambiguously construct an AND-NOT network, which implies that all algorithms or results can be stated at the “wiring diagram level.”

2 Definitions

Definition 1. For a signed directed graph \( G = (V_G, E_G) \), we denote \( I_i = \{ j : (j, i) \in E_G \} \), \( I_i^- = \{ j : (j, i, +) \in E_G \} \) and \( I_i^+ = \{ j : (j, i, -) \in E_G \} \). That is, \( I_i \) is the set of all incoming edges for node \( i \), and \( I_i^+ \), resp. \( I_i^- \) is the subset of positive, resp. negative, edges. All graphs in the rest of the paper will be signed directed graphs unless noted otherwise.

In order to simplify the graphical representation, we denote two negative (positive) edges between \( i \) and \( j \) by a bidirectional negative (positive) edge, \( \xrightarrow{-} - \xleftarrow{+} \). If the edges have different signs we denote them by \( \xrightarrow{-} \xleftarrow{-} \).

Definition 2. An AND-NOT function is a Boolean function, \( h : \{0, 1\}^n \rightarrow \{0, 1\} \), such that \( h \) can be written in the form

\[
h(x_1, \ldots, x_n) = \bigwedge_{j \in P} x_j \wedge \bigwedge_{j \in N} \neg x_j,
\]

where \( P \cap N = \{ \} \). If \( P = N = \{ \} \), then \( h \) is the constant function \( 1 \). If \( i \in P \) (\( i \in N \), respectively) we say that \( i \) or \( x_i \) is a positive (negative) regulator of \( h \) or that it is an activator (repressor). An AND-NOT network is a Boolean network \( (G, f) \), \( f = (f_1, \ldots, f_n) : \{0, 1\}^n \rightarrow \{0, 1\}^n \), such that \( f_i \) is an AND-NOT function for all \( i = 1, \ldots, n \). AND-NOT networks are also called signed conjunctive networks.

Definition 3. The wiring diagram of an AND-NOT network is defined by a graph \( G = (V_G, E_G) \) with vertices \( V_G = \{1, \ldots, n\} \) (or \( \{x_1, \ldots, x_n\} \)) and edges \( E_G \) given as follows: \((i, j, +) \in E_G \) if \( i \in N \) and \( j \in N \) or \( i \in N \) and \( j \in P \). Notice that nodes corresponding to constant functions have in-degree zero. Also, the wiring diagram of an AND-NOT network contains all the information about the network; that is, we only need to specify the wiring diagram in order to define an AND-NOT network.

Example 1. Consider the Boolean network \( f = (f_1, \ldots, f_6) : \{0, 1\}^6 \rightarrow \{0, 1\}^6 \) given by \( f(x) = (x_1 \wedge x_4 \wedge \neg x_5 \wedge x_6 \wedge \neg x_3 \wedge \neg x_1 \wedge \neg x_3 \wedge x_6 \wedge \neg x_1, 1) \). It is easy to see that \( f \) is an AND-NOT network. Its wiring diagram is shown in Figure 1.

![Fig. 1: Wiring diagram of the AND-NOT network in Example 1.](image-url)
3 Results

In this section we show why AND-NOT networks are a good framework for modeling biological systems.

3.1 AND-NOT networks are general enough for modeling

One issue that can potentially arise when only using certain classes of networks is that one can have difficulty in modeling certain processes. For example, the family of AND networks does not allow modeling negative interactions. Another example is that the family of linear networks, does not allow modeling signed interactions. In order for a family of networks to be useful for modeling, it has to allow modeling any type of interaction.

Here we show that the family of AND-NOT networks is general enough for modeling. More precisely, we show that for any finite dynamical system, there exists an AND-NOT network (possibly with more nodes) such that they share key dynamical properties.

Theorem 1. Let \( h = (h_1, \ldots, h_n) : S \to S \) be a finite dynamical system, where \( S = X_1 \times \cdots \times X_d \) and all \( X_i \)'s are finite. Then, there exists an AND-NOT network \( g : \{0,1\}^m \to \{0,1\}^m \) such that there is a bijection between the steady states of \( h \) and \( g \). Furthermore, \( g \) and the bijection between steady states is given algorithmically. We say that \( g \) is an AND-NOT representation of \( h \).

Proof. A simple proof uses the facts that any finite dynamical system can be written as a Boolean network [31], and that any Boolean function has a conjunctive normal form.

In [31], the authors proved algorithmically that for any finite dynamical system \( h \), there exists a Boolean network \( f \) (possibly with more nodes) such that \( h \) and \( f \) have the same number of steady states. Furthermore, the bijection of steady states is also given algorithmically. Therefore, we only need to show that there exists and

AND-NOT network \( g \), such that there is a bijection between the steady states of \( f \) and \( g \).

We proceed by induction. First, consider the conjunctive normal form of \( f_0 \): \( f_0 = w_1 \wedge w_2 \wedge \cdots \wedge w_r \), where \( w_j \) is of the form \( w_j(x) = s_1 x_1 \lor s_2 x_2 \lor \cdots \lor s_n x_n \) with \( s_i \in \{0,1\} \) (\( id = \)identity function). Notice that \( \neg w_j \) is an AND-NOT function. Then, define the BN \( k = (k_1, \ldots, k_{n+r}) : \{0,1\}^{n+r} \to \{0,1\}^{n+r} \) in variables \( x_1, \ldots, x_n, y_1, \ldots, y_r \) by \( k_i(x,y) = f_i(x) \) for \( i = 1, \ldots, n \) and \( k_{n+i}(x,y) = \neg y_i \land \neg y_j \land \cdots \land \neg y_r \) and \( k_{n+r}(x,y) = \neg w_i(x) \) for \( i = n+1, \ldots, n+r \).

We now check that the function \( \phi(x) = (x, \neg w_1(x), \ldots, \neg w_r(x)) \) gives a one-to-one correspondence between steady states of \( f \) and \( k \). Suppose that \( f(x) = x \), then

\[
\phi(f(x)) = \phi(x, \neg w_1(x), \ldots, \neg w_r(x)) = k(f_1(x), \ldots, f_{n-r}(x), w_1(x)) = (x, \neg w_1(x), \ldots, \neg w_r(x)) = \phi(x); \text{ that is, } \phi(x) \text{ is a steady state of } k. \text{ Now, suppose that } k(x,y) = (x,y) \text{ and notice that in this case } y_i = k_i(x,y) = \neg w_i(x) \text{; then } (x,y) = \phi(x).
\]

Also, \( f(x) = (f_1(x), \ldots, f_{n-r}(x), f_n(x)) = (k_1(x,y), \ldots, k_{n-r}(x,y), w_1(x)) = (x_1, \ldots, x_{n-1}, \neg y_1 \land \neg y_2 \land \cdots \land \neg y_r) = (x_1, \ldots, x_{n-1}, k_n(x,y)) = (x_1, \ldots, x_{n-1}, x_n) = x \).

That is, \( x \) is a steady state of \( f \). Therefore, \( k = (k_1, \ldots, k_{n+r}) \) is a BN where \( k_1, \ldots, k_{n+r} \) are AND-NOT functions and such that there is a one-to-one correspondence between the steady states of \( f \) and \( k \). By induction, it follows that there is an AND-NOT network \( g : \{0,1\}^m \to \{0,1\}^m \) together with a bijection between the steady states of \( f \) and \( g \).

Therefore, there is a bijection between the steady states of \( h \) and \( g \). Furthermore, \( g \) and the bijection are given algorithmically.

The transformation of finite dynamical systems to Boolean networks has been discussed in [31]. So, in the rest of the paper we will focus on Boolean networks and AND-NOT networks.

Example 2. Consider the BN \( f : \{0,1\}^5 \to \{0,1\}^5 \) given by \( f_1 = x_2 \lor \neg x_4, f_2 = x_1 \land x_3, f_3 = (x_2 \lor \neg x_4) \land x_5, f_4 = x_3 \lor x_5, f_5 = x_7 \). The wiring diagram of \( f \) is given in Figure 2 (left). In order to transform this BN to an AND-NOT network we introduce the variable \( x_6 \) with Boolean function \( f_6 = \neg x_2 \land \neg x_4 \) and \( f_7 = \neg x_3 \land \neg x_4 \). Variables \( x_6 \) and \( x_7 \) will be used in \( g_1 \) and \( g_2 \). Notice that since \( x_2 \lor \neg x_4 \) appears again in \( f_3 \), we can simply reuse \( x_6 \) to keep the number of extra variables small as possible. Then the AND-NOT network is \( g : \{0,1\}^7 \to \{0,1\}^7 \) given by \( g_1 = \neg x_6, g_2 = x_1 \land x_3, g_3 = \neg x_6 \land x_5, g_4 = \neg x_7, g_5 = x_3, g_6 = \neg x_2 \land x_4, g_7 = \neg x_3 \land \neg x_5 \). The wiring diagram of \( g \) is shown in Figure 2 (right).

An additional step in the transformation that can keep the number of extra variables small is given by the following proposition.

© 2013 NSP
Natural Sciences Publishing Corp.
we will now show that in practice we can use Proposition 1 to transform the BN into a AND-NOT network. Notice that the effect of this transformation on the wiring diagram is simple, we simply change the signs of the edges around node 2.

Proposition 1. Let \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) be a BN and define \( g : \{0,1\}^n \rightarrow \{0,1\}^n \) by \( g = N_k \circ f \circ N_k \), where \( N_k(x_1, \ldots, x_n) = (x_1, \ldots, x_{k-1}, \neg x_k, x_{k+1}, \ldots, x_n) \). Then \( f \) and \( g \) are dynamically equivalent.

Proof. It is enough to notice that \( N_k \) is invertible with inverse \( N_k \). Then, \( g' = N_k \circ f' \circ N_k \); that is, evaluating \( f \) is equivalent to evaluating \( g \).

If some functions of a BN are OR-NOT functions, then we can use Proposition 1 to transform the BN into a BN in the same number of variables such that the OR-NOT functions become AND-NOT functions. Also, Proposition 1 can be used to transform constant functions \( f_k = 0 \) into constant functions \( f_k = 1 \) (if \( f_k = 0 \), then the \( k \)-th coordinate function of \( N_k \circ f \circ N_k \) is the constant function 1).

Example 3. Consider the BN \( f : \{0,1\}^3 \rightarrow \{0,1\}^3 \) given by \( f_1 = x_2, f_2 = x_1 \lor \neg x_3, f_3 = x_2 \land x_3 \). The wiring diagram of \( f \) is in Figure 3 (left). Since \( f_2 \) is an OR-NOT function, we can transform it to a AND-NOT function using Proposition 1. Consider \( g : N_2 \circ f \circ N_2 \), given by \( g(x) = N_2(f((x_1, \neg x_2, x_3)) = N_2(\neg x_2, x_1 \lor \neg x_3, \neg x_2 \land x_3) = (\neg \neg x_2, \neg x_1, \lor \neg x_3, \neg x_2 \land x_3) \), with wiring diagram shown in Figure 3 (right). Then, \( f \) is dynamically equivalent to an AND-NOT network. Notice that the effect of this transformation on the wiring diagram is simple, we simply change the signs of the edges around node 2.

Table 1: Number of extra variables introduced by the AND-NOT representation. The number of nodes of \( f \) and its AND-NOT representation, \( g \), are denoted by \( n \), \( m \), respectively. The BNs were taken from [34, 35, 36, 37, 38].

<table>
<thead>
<tr>
<th>( n )</th>
<th>( m )</th>
<th>% increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>13</td>
<td>8%</td>
</tr>
<tr>
<td>12</td>
<td>15</td>
<td>20%</td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>7%</td>
</tr>
<tr>
<td>20</td>
<td>24</td>
<td>20%</td>
</tr>
<tr>
<td>23</td>
<td>26</td>
<td>13%</td>
</tr>
<tr>
<td>28</td>
<td>28</td>
<td>0%</td>
</tr>
<tr>
<td>40</td>
<td>43</td>
<td>7.5%</td>
</tr>
</tbody>
</table>

In order to study this question, we have applied the procedure to several published models in the literature and studied the question using randomly generated Boolean networks. The first study shows that the increase in the number of variables for published models is modest (Table 1). The number of variables was increased by 14% on average with a maximum value of 4 extra nodes. In order to determine the number of extra nodes introduced by our algorithm for more general BNs, we did a statistical analysis. To mimic wiring diagrams coming from biological systems, the edges followed a power law distribution and we considered the maximum in-degree less than or equal to \( K \) for \( K = 1, \ldots, 10 \) (see Appendix A.
Table 2: Average number of extra variables introduced by the AND-NOT representation for random BNs.

<table>
<thead>
<tr>
<th>in-deg ≤ K</th>
<th>% increase</th>
</tr>
</thead>
<tbody>
<tr>
<td>K = 1, 2</td>
<td>0%</td>
</tr>
<tr>
<td>K = 3</td>
<td>5.2%</td>
</tr>
<tr>
<td>K = 4</td>
<td>10.8%</td>
</tr>
<tr>
<td>K = 5</td>
<td>16.2%</td>
</tr>
<tr>
<td>K = 6</td>
<td>20.8%</td>
</tr>
<tr>
<td>K = 7</td>
<td>24.8%</td>
</tr>
<tr>
<td>K = 8</td>
<td>28.6%</td>
</tr>
<tr>
<td>K = 9</td>
<td>32.3%</td>
</tr>
<tr>
<td>K = 10</td>
<td>36.1%</td>
</tr>
</tbody>
</table>

for details). The results of this second study are shown in Table 2. For example, all networks where nodes have in-degree bounded by \( K = 2 \) can be transformed to AND-NOT networks without increasing the number of nodes. For networks where nodes have in-degree bounded by \( K = 4 \), our method increases the number of nodes by 10.8% on average (see Appendix A for details). It is important to mention that in both tables, the growth in the number of extra nodes is far less than exponential.

3.3 AND-NOT networks can be useful in theoretical analysis

As mentioned in the Introduction, the specialized logic of AND-NOT networks can be used to obtain better theoretical results. Such results can arise directly (e.g. [5, 20]) or by applying results about general Boolean networks to the family of AND-NOT networks. In this section we show examples of the latter. First, we need the following definitions.

Let \( C \) be a feedback loop of a graph \( G \). We say that \( C \) is a strong feedback loop if there are no edges of the form \( k \rightarrow i, k \rightarrow j \) in \( G \setminus C \) such that \( i, j \in C \).

For example, consider the graph \( G \) in Figure 4. The feedback loop \( \{3, 4\} \) is not strong because of the edges \( 1 \rightarrow 3, 1 \rightarrow 4; \) \( \{5, 6\} \) and \( \{1, 2, 4, 3, 5\} \) are not strong because of the edges \( 1 \rightarrow 5, 1 \rightarrow 4 \). All other feedback loops are strong.

Our first result in this section is an application of [39, Theorem 3.2] to the family of AND-NOT networks (see Appendix B for the proof).

Theorem 2. Let \( W \) be the wiring diagram of an AND-NOT network, and suppose \( J \) intersects all strong positive feedback loops of \( W \). Then, the number of steady states is at most \( 2^{|J|} \).

Example 4. Consider the AND-NOT network with wiring diagram given in Figure 4. The only strong positive feedback loops are \( \{1, 2\} \) and \( \{1, 3, 5\} \). Since \( J = \{1\} \) intersects them, Theorem 2 guarantees that there are at most \( 2^{|J|} = 2 \) steady states.

Intuitively, Theorem 2 is telling us which positive feedback loops contribute to the presence of steady states; it says that they have to be strong. We also provide a slight generalization of Theorem 2. We need the following definition.

A feedback loop \( C \) of a graph \( W \) is called inconsistent if there is a vertex \( k_C \) such that there is a positive path of the form \( k_C \rightarrow i_1 \rightarrow \cdots \rightarrow i_r \rightarrow t_C \) from \( k_C \) to \( t_C \in C \) and a negative path of the form \( k_C \rightarrow j_1 \rightarrow \cdots \rightarrow j_r \rightarrow u_C \) from \( k_C \) to \( u_C \in C \) such that \( k_C \rightarrow t_C, k_C \rightarrow u_C \) are not edges in \( C \) and \( |I_{j_1}| = \cdots = |I_{j_r}| = 1 \). When such vertex \( k_C \) does not exist, we say that \( C \) is consistent.

For example, consider the graph \( W \) in Figure 5. The positive feedback loop \( \{3, 4\} \) is inconsistent because of the paths \( 1 \rightarrow 3 \) and \( 1 \rightarrow 2 \). The positive feedback loop \( \{5, 6\} \) is inconsistent because of the paths \( 1 \rightarrow 3 \rightarrow 5 \) and \( 1 \rightarrow 6 \). Also, the positive feedback loop \( \{1, 2, 4, 3, 5\} \) is inconsistent because of the paths \( 1 \rightarrow 3 \rightarrow 5 \) and \( 1 \rightarrow 2 \). Then, the only consistent feedback loops are \( \{1, 2\} \) and \( \{1, 3, 5\} \).

![Fig. 4: Graph with only one strong positive feedback loop.](image)

![Fig. 5: Wiring diagram of the AND-NOT network in Example 5.](image)
Theorem 3. Let $W$ be the wiring diagram of an AND-NOT network, and suppose $J$ dominates $W$. Then the number of steady states is at most $2^{|J|}$.

It is not difficult to see that the bound given by Theorem 2 is greater than or equal than the bound given by Theorem 3. The next example shows that the inequality is in some cases strict.

Example 5. Consider the BN $f: \{0,1\}^6 \rightarrow \{0,1\}^6$ given by

\begin{align*}
f_1 &= x_2 \land x_5, \\
f_2 &= x_1, \\
f_3 &= x_1 \land \neg x_4, \\
f_4 &= \neg x_2 \land \neg x_3, \\
f_5 &= x_3 \land \neg x_6, \\
f_6 &= \neg x_1 \land \neg x_5,
\end{align*}

Its wiring diagram is shown in Figure 5. It is easy to see that $\{1,3,5\}$ intersects all strong positive feedback loops. Then, Theorem 2 gives the upper bound $2^3 = 8$. On the other hand, since $\{1\}$ dominates the wiring diagram, Theorem 3 gives the upper bound 2. That is, Theorem 3 gave a better upper bound on the number of steady states. Notice that in this case the actual number of steady states is 2, namely, 000101 and 111010.

One might argue that having better results for AND-NOT networks is not enough to justify their use. After all, since we are considering a smaller family of Boolean networks we should of course obtain stronger results. However, the combination of Theorem 1 and results about AND-NOT networks automatically generates theorems for all Boolean networks. Furthermore, such combination can in some cases provide stronger results. This deserves further explanation which is illustrated in Figure 6. Consider a theorem about Boolean networks that gives us information about certain dynamical properties, “Thm.”. On the other hand, consider a similar theorem about AND-NOT networks, “Thm.”. Then, given a Boolean network $f$, we have two choices, we can apply Thm. to $f$; or, we can use Theorem 1 to find the AND-NOT representation of $f$, then apply Thm., and then use Theorem 1 to obtain information about the original Boolean network $f$. In Section 3.4 we use a published Boolean model to show that the latter can give stronger results.

For example, combining Theorem 1 and 3 we obtain the following theorem.

Theorem 4. Let $f$ be any Boolean network and suppose that $J$ dominates the wiring diagram of its AND-NOT representation. Then, $f$ has at most $2^{|J|}$ steady states.

We now show that this theorem can in fact provide a better upper bound for the number of steady states.

3.4 Application to Th-cell differentiation

We apply our results to the BN model proposed in [36] for Th-cell differentiation. The model is a BN in 23 variables, $f: \{0,1\}^{23} \rightarrow \{0,1\}^{23}$. Below is the list of Boolean functions. The wiring diagram is shown in Figure 7.

\begin{align*}
x_1 &= GATA3, & f_1 &= (x_1 \lor x_{21}) \land \neg x_{22}; \\
x_2 &= IFN - \beta, & f_2 &= 0; \\
x_3 &= IFN - \beta, & f_3 &= x_2; \\
x_4 &= IFN - \gamma, & f_4 &= (x_{14} \lor x_{16} \lor x_{20} \lor x_{22}) \land \neg x_{19}; \\
x_5 &= IFN - \beta R, & f_5 &= x_4; \\
x_6 &= IL - 10, & f_6 &= x_{11}; \\
x_7 &= IL - 10 R, & f_7 &= x_{36}; \\
x_8 &= IL - 12, & f_8 &= 0; \\
x_9 &= IL - 12 R, & f_9 &= x_{38} \lor \neg x_{21}; \\
x_{10} &= IL - 18, & f_{10} &= 0; \\
x_{11} &= IL - 18 R, & f_{11} &= x_{10} \lor \neg x_{21}; \\
x_{12} &= IL - 4, & f_{12} &= x_1 \lor \neg x_{18}; \\
x_{13} &= IL - 4 R, & f_{13} &= x_{12} \lor \neg x_{17}; \\
x_{14} &= IRAK, & f_{14} &= x_{11}; \\
x_{15} &= JAK1, & f_{15} &= x_{38} \land \neg x_{17}; \\
x_{16} &= NFAT, & f_{16} &= x_{23}; \\
x_{17} &= SOCS1, & f_{17} &= x_{18} \lor x_{22}; \\
x_{18} &= STAT1, & f_{18} &= x_3 \lor x_{15}; \\
x_{19} &= STAT3, & f_{19} &= x_{37}; \\
x_{20} &= STAT4, & f_{20} &= x_9 \lor \neg x_1; \\
x_{21} &= STAT6, & f_{21} &= x_{31}; \\
x_{22} &= T - bet, & f_{22} &= (x_{14} \lor x_{22}) \land \neg x_1; \\
x_{23} &= TCR, & f_{23} &= 0.
\end{align*}

Using our algorithms we obtain the AND-NOT network, $g: \{0,1\}^{26} \rightarrow \{0,1\}^{26}$, shown in Figure 8. It turns out that the set $\{1,22\}$ dominates the wiring diagram of $g$ (see Appendix C for details). Then, by Theorem 4, the number of steady states of $f$ is at most $2^2 = 4$. On the other hand, all previous results about steady states (e.g. [8, 19]) give 8 as the upper bound. That is, using the AND-NOT representation can provide a better upper bound, even for general Boolean networks. The actual number of steady states of the model is 3 (see [36] for details).

4 Discussion

The results presented in this paper, together with other results in the literature, support that the family of AND-NOT networks are general enough for modeling and simple enough for theoretical analysis. Given any
finite dynamical system, it is possible to create an AND-NOT network such that they have similar dynamical properties. This has two implications: First, this means that using AND-NOT networks in modeling does not pose any technical restriction on the type of interactions one can model. Second, every result about AND-NOT networks can be reformulated as questions about graphs; then, one can use tools from graph theory and combinatorics to study them (e.g. antichains, posets, inclusion-exclusion principle, independent sets [5, 20]). This deserves further investigation.

Finally, we point out that AND-NOT networks are special cases of so-called nested canalyzing Boolean networks. They are Boolean networks where each Boolean function can be written in the form $h = y_i \lor \bigwedge (y_j \lor \bigvee (y_k \lor \bigwedge \ldots ))$, where $i \in \{1, \ldots, n\}$ (n is the number of nodes), $\bigwedge j \in \{\land, \lor\}$, and $y_j \in \{x_j, \neg x_j\}$. These were first introduced in [10, 11] as good candidates for models with “biologically meaningful” regulatory rules, and have since been studied extensively. In [13] this concept was generalized to multi-state models, and it was shown there that the large majority of regulatory rules that appear in published models of biological networks are of this form. It was shown furthermore that nested canalyzing networks have dynamic properties one would expect to find in biological networks, such as short limit cycles and a small number of attractors. Thus, the results in the present paper imply that in order to study the steady state behavior of general network models, one can focus on the very restrictive class of nested canalyzing networks [14], instantiated as AND-NOT networks and make use of their very special properties.

### Appendix A

We describe here the details of the study to determine how many nodes are added by the construction of the AND-NOT representation. To mimic wiring diagrams coming from biological systems, the edges followed a power law distribution. More precisely, given $K$ fixed and a parameter $\gamma$, the probability for a node to have $k < K$ nodes is $p_k = k^{-\gamma}$ (up to a normalization factor). For example, if $K = 4$, the probabilities of having 1, 2, 3 and 4 nodes are $p_1 = c1^{-\gamma}$, $p_2 = 2c2^{-\gamma}$, $p_3 = 3c3^{-\gamma}$ and $p_4 = 4c4^{-\gamma}$, respectively, where $c = \frac{1}{1^{-\gamma} + 2^\gamma + 3^\gamma + 4^\gamma}$ so that $p_1 + p_2 + p_3 + p_4 = 1$. Also, to mimic biological regulation, we restricted our analysis to Boolean functions that admitted a sign assignment for the edges. These Boolean functions are called unate, biologically meaningful and regulatory functions [7, 9, 8].

Denote with $e_k$ the average number of extra nodes introduced by a Boolean function in $k$ variables. Then, a BN that follows the distribution mentioned above will
have, on average, \( p_1 e_1 + p_2 e_2 + \cdots + p_k e_k \) extra nodes. Now, we need to estimate \( e_k \).

Consider a Boolean function, \( h \), that depends on \( k \) variables. For \( k = 1 \) there are 2 functions, \( h = x_1 \) and \( h = \neg x_1 \) and we do not need to introduce any new nodes; then \( e_1 = 0 \). For \( k = 2 \) there are 8 functions and they are of the form \( h = s_1 x_1 \lor s_2 x_2 \) or \( h = s_1 x_1 \lor s_2 x_2 \), where \( s_1 x_1 = x_1 \) or \( s_1 x_1 = \neg x_1 \). For functions of the form \( h = s_1 x_1 \lor s_2 x_2 \) we do not introduce any new nodes, and for functions of the form \( h = s_1 x_1 \lor s_2 x_2 \) we can use Proposition 1 to transform \( h \) to an AND-NOT function, so we do not introduce new nodes either. Then \( e_2 = 0 \). For \( k = 3 \), there are 72 functions. An exhaust-internal search analysis shows that of those 72 Boolean functions, 16 of extra nodes in this case is determined. For \( k = 4 \), there are 1824 functions. An exhaust-internal search analysis shows that of those 1824 functions, 32 introduce 0 nodes, 320 introduce 1 node, 48 introduce 2 nodes, 960 introduce 3 nodes and 32 introduce 4 nodes; thus the average number of extra nodes in this case is determined.

Theorem A.1. The average number of extra nodes for a unate function of \( k \) variables is at most \( C(k, \lfloor k/2 \rfloor) \); that is, \( e_k \leq C(k, \lfloor k/2 \rfloor) \). Where \( C \) is the binomial coefficient and \( \lfloor \rfloor \) is the floor function.

Proof. Without loss of generality we assume the CNF of the Boolean function \( f \) has no negative signs. Let \( f = w_1 \land \cdots \land w_r \) be the CNF, where \( w_1 \) has the form \( w_1 = x_1 \lor \cdots \lor x_i \). For each \( i \), define \( S_i = \{ l : x_l \text{ appears in } w_i \} \).

Now, if \( i, j \) such that \( S_i \subseteq S_j \), then we can simplify \( w_1 \lor w_j \) to \( w_i \) (e.g. \( x_1 \lor x_2 \land (x_1 \lor x_2) \lor x_3 = x_1 \lor x_2 \)). That is, we can simplify the CNF so that \( S_i \nsubseteq S_j \) for all \( i \neq j \).

Thus, \( S_1, \ldots, S_r \) is a family of subsets of \( \{1, \ldots, k\} \) such that no one is contained in the other. Sperner's theorem [41] states that \( r \leq C(k, \lfloor k/2 \rfloor) \). This implies that for any unate function in \( k \) variables, we need at most \( C(k, \lfloor k/2 \rfloor) \) extra nodes to obtain the AND-NOT representation. Therefore, \( e_k \leq C(k, \lfloor k/2 \rfloor) \).

It is important to mention that the exhaustive-search analysis done for \( k = 3, 4, 5 \) suggests that \( e_k \) is actually much smaller than \( C(k, \lfloor k/2 \rfloor) \). In fact, we did a statistical analysis for \( k = 6, \ldots, 10 \) using a total of 5000000 Boolean functions chosen at random (100000 for each \( k \)). The analysis shows the following approximations:

\[
\begin{align*}
e_6 &\approx 5.32, e_7 \approx 7.04, e_8 \approx 9.32, e_9 \approx 12.24, e_{10} \approx 15.96.
\end{align*}
\]

Table 2 shows a summary of our analysis for \( \gamma \approx 2.5 \). For example, if \( K = 4 \), then the fractions of functions with \( 1, 2, 3 \) and 4 variables are functions of which \( p_1 = 0.786 \), \( p_2 = 0.139 \), \( p_3 = 0.0504 \) and \( p_4 = 0.0246 \), respectively.

Then, the average number of extra nodes is

\[
100(p_1 e_1 + p_2 e_2 + p_3 e_3 + p_4 e_4) = 100(0.786 \times 0 + 0.139 \times 0 + 0.0504 \times 1 + 0.0246 \times 2.35) \approx 10.8%.
\]

Appendix B

Here we prove Theorem 2 and 3. As mentioned in Section 3.3, Theorem 2 is an application of [39, Theorem 3.2] to the family of AND-NOT networks. First we need the following definition.

Let \( f : \{0,1\}^n \to \{0,1\}^n \) be a Boolean network and consider \( x \in \{0,1\}^n \). Then, \( W(x) = (V, E) \) is the graph with vertices \( V = \{1, \ldots, n\} \) and the following edges:

\((i, i, +) \in E \) if \( x_i = 0 \) and \( f_i(x) < f_i(x + e_i) \), or if \( x_i = 1 \) and \( f_i(x - e_i) < f_i(x) \);

\((j, i, -) \in E \) if \( x_j = 0 \) and \( f_j(x) > f_j(x + e_j) \), or if \( x_j = 1 \) and \( f_j(x - e_j) > f_j(x) \);

Thus, \( e_i \) is the vector given by \( (e_i)_j = \delta_{ij} \delta \) (where \( \delta \) is the Kronecker delta). Notice that if \((j, i, +) \) or \((j, i, -) \) is an edge in \( W(x) \), then changing the \( j \)-th coordinate of \( x \) produces a change in \( f_j \). Notice that for AND-NOT networks we have that \( W(x) \subseteq W \) for all \( x \); in fact, this is true for many more general networks.

Theorem B.1 [39] Let \( f \) be a Boolean network and suppose \( a \) and \( b \) are steady states of \( f \). Then, there exists \( x \) such that \( W(x) \) has a positive feedback loop with vertices in the set \( \{i : a_i \neq b_i\} \).

We now prove Theorem 2.

Proof. Let \( f : \{0,1\}^n \to \{0,1\}^{|J|} \) be a Boolean network defined by \( \phi(x) = x_J \). We will show that if \( a \neq b \) are steady states of \( g \), then \( \phi(a) \neq \phi(b) \). Consider \( a \neq b \) steady states of \( g \); then, by Theorem B.1., there exists \( x \) such that \( W(x) \) has a positive feedback loop, \( C \), with vertices in the set \( \{i : a_i \neq b_i\} \).

We claim that \( C \) is a strong positive feedback loop of \( W \). By contradiction, suppose there is \( k \in \{1, \ldots, n\} \) and \( i, j \in C \) such that \( k \to i \) and \( k \to j \) are edges in \( W(x) \) but not in \( C \). Then, \( W(x) \) has edges of the form \((l_1, i, \pm) \) and \((l_2, j, \pm) \) where \( l_1, l_2 \neq k \). On the other hand, since \((k, i, +) \), \((k, j, -) \in C \subseteq W(x) \subseteq W \), we have that \( f_k = x_k \) and \( f_j = \neg x_k \). We have two cases \( x_k = 0 \) or \( x_k = 1 \). In the case \( x_k = 0 \) we obtain that \( f_j = 0 \) for all values of \( x_1, \ldots, x_k-1, x_k+1, \ldots, x_n \). In particular, \( W(x) \) cannot have an edge of the form \((l, i, \pm) \) with \( i \neq k \); this is a contradiction. In the case \( x_k = 1 \) we obtain that \( f_j = 0 \) for all values of \( x_1, \ldots, x_k-1, x_k+1, \ldots, x_n \). In particular, \( W(x) \) cannot have an edge of the form \((l, j, \pm) \) with \( l \neq k \); this is a contradiction as well. Therefore, \( C \) is strong.

Since \( C \) is a strong positive feedback loop in \( W \), \( C \) must intersect \( J \). Since \( C \) has all its vertices in the set \( \{i : a_i \neq b_i\} \), \( J \) intersects the set \( \{i : a_i \neq b_i\} \). Therefore \( \phi(a) = a_j \neq b_j = \phi(b) \). It follows that the restriction of \( \phi \) to the set of steady states is an injective function. Therefore, \( \{x : f(x) = x\} \subseteq \{|0,1|\} = 2^{J} \).
It is important to mention that Theorem 2 was also proven in [19] using different techniques.

We now prove Theorem 3.

Proof. Let \( f : \{0,1\}^n \rightarrow \{0,1\}^n \) be an AND-NOT network with wiring diagram \( W \). Let \( C \) be a positive feedback loop that is strong and inconsistent. Then, there is a vertex \( k \in C \) such that there is a positive path of the form \( k_C \rightarrow i_1 \rightarrow \ldots \rightarrow i_t \rightarrow k \). From \( k_C \) to \( i_t \in C \), there is a negative path of the form \( k_C \rightarrow j_1 \rightarrow \ldots \rightarrow j_r \rightarrow u_C \) from \( k_C \) to \( u_C \). Suppose that \( k_C \rightarrow i_t \rightarrow i_{t+1} \rightarrow \ldots \rightarrow i_s \rightarrow k \) is inconsistent (and strong) in \( C \) for all \( i \in I_t \). Let \( G \) be the graph obtained by adding to \( W \) all edges of the form \( k_C \rightarrow i_t \) and \( k_C \rightarrow u_C \). We claim that the steady states of \( f \) and \( g \) are the same. We prove this by induction on the number of extra edges.

Suppose that \( W \) and \( Z \) only differ in the edge \( k \rightarrow t \), then, by definition we must also have a path \( k \rightarrow i_1 \rightarrow \ldots \rightarrow i_t \rightarrow t \). Suppose that \( g(x) = x \), we need to show that \( f_j(x) = x_j \) for all \( j \). Since \( W \) and \( Z \) only differ in the edge \( k \rightarrow t \) we have \( f_j(x) = g(t) \) for \( j \neq t \), \( g_j = f_j \land x_k \) and \( f_j = x_i \land \ldots \). Then, \( f_j(x) = g(x) = x_k \) for \( j \neq t \). It remains to show that \( f_t(x) = x_t \). Consider first the case \( x_t = 0 \), then \( g_t(x) = 0 \) and \( x_0 = 0 \) for some \( i \in I_t \). If \( i \neq k \), we have that the edge \( i \rightarrow t \) is in \( W \) and \( f_i = x_0 \land x_i \lor \ldots \); then, \( f_t(x) = x_t \land 0 \land \ldots = 0 \). If \( i = k \), then \( x_0 = 0 \) which implies that \( x_i = 0 \) (because of the edge \( k \rightarrow i_t \)); similarly, we obtain that \( x_i = 0 \). Then, \( f_t(x) = x_t \). Now consider the case \( x_t = 1 \). Since \( 1 = x_t = g_t(x) = f_t(x) \land x_k \), we have \( f_t(x) = 1 = x_t \). A similar argument shows that if \( f(x) = x \), then \( g(x) = x \). The proof for when \( W \) and \( Z \) only differ in the edge \( k \rightarrow t \) is analogous. By induction we obtain that \( f \) and the AND-NOT network obtained by a completion of \( W \) have the same steady states.

Now, we claim that \( J \) intersects all strong positive feedback loops of \( Z \). Let \( C' \) be a strong positive feedback loop of \( Z \). Then we have two cases: \( C' \) is in \( W \) or it is not. Consider the case \( C' \subseteq W \). Then, \( C' \) is a strong positive feedback loop in \( W \). If \( C' \) is consistent in \( W \), then it intersects \( J \). If \( C' \) is inconsistent (and strong) in \( W \), then it also intersects \( J \). Now consider the case \( C' \not\subseteq W \). Then, at least one edge of \( C' \) is of the form \( k_C \rightarrow i_t \) or \( k_C \rightarrow u_C \) for some \( C \) strong and inconsistent that does not intersect \( J \). Then, \( k_C \in J \) and \( J \) intersects \( C \). In any case we obtain that \( J \) intersects all strong positive feedback loops of \( Z \).

Then, the number of steady states of \( g \), and hence \( f \), is at most \( 2^{2j} \).

Appendix C

We first analyze the original BN using previous results. In [36], the authors showed that the positive feedback loops of the BN \( f : \{0,1\}^{23} \rightarrow f : \{0,1\}^{23} \) are:

\[ \{4, 5, 15, 18, 12, 13, 21, 11, 14\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 9, 20\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 1, 6, 7, 19\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 1, 20\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 1, 22\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 11, 14\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 9, 20\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 1, 6, 7, 19\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 1, 20\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 1, 22\} \]

\[ \{4, 5, 15, 18, 22\} \]

\[ \{4, 5, 15, 18, 22, 17, 13, 21, 11, 14\} \]

\[ \{4, 5, 15, 18, 22, 17, 13, 21, 9, 20\} \]

\[ \{4, 5, 15, 18, 22, 17, 13, 21, 1, 6, 7, 19\} \]

\[ \{4, 5, 15, 18, 22, 17, 13, 21, 1, 20\} \]

\[ \{4, 5, 15, 18, 22, 1, 12, 13, 21, 11, 14\} \]

\[ \{4, 5, 15, 18, 22, 1, 12, 13, 21, 9, 20\} \]

\[ \{4, 5, 15, 18, 22, 1, 6, 7, 19\} \]

\[ \{4, 5, 15, 18, 22, 1, 20\} \]

\[ \{12, 13, 21, 1\} \]

\[ \{13, 21, 1, 22, 17\} \]

\[ \{22\} \]

\[ \{22, 1\} \]

We will use the following two theorems (proven in [8, 19], respectively) that give upper bounds on the number of steady states.

**Theorem 5.** Let \( W \) be the wiring diagram of a BN network and suppose \( J \) is a set of vertices that intersects all positive feedback loops in \( W \). Then, the number of steady states is at most \( 2^{|J|} \).

**Theorem 6.** Let \( W \) be the wiring diagram of a BN network and suppose \( J \) is a set of vertices that intersects all functional positive feedback loops in \( W \). Then, the number of steady states is at most \( 2^{|J|} \).

It is easy to see that all positive feedback loops intersect the set \( \{1, 4, 22\} \). Therefore, Theorem 5 gives the upper bound \( 2^3 = 8 \). Also, it is possible to show that the functional positive feedback loops are \( \{4, 5, 15, 18, 12, 13, 21, 11, 14\} \), \( \{22\} \) and \( \{1\} \) (e.g. using the GINsim software [42]). Therefore, Theorem 6 gives the upper bound 8 as well.

We now analyze the AND-NOT network using our results. The positive feedback loops of the AND-NOT network in Figure 8 are the following (new nodes are in bold):

\[ \{4, 5, 15, 18, 12, 13, 21, 11, 14\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 9, 20\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 6, 1, 6, 7, 19\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 6, 1, 20\} \]

\[ \{4, 5, 15, 18, 12, 13, 21, 6, 1, 22\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 11, 14\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 9, 20\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 6, 1, 6, 7, 19\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 6, 1, 20\} \]

\[ \{4, 5, 15, 18, 17, 13, 21, 6, 1, 22\} \]
Those feedback loops that contain 4 and 13 are inconsistent because of the paths 1 \rightarrow 12 \rightarrow 13, 1 \rightarrow 6 \rightarrow 7 \rightarrow 19 \rightarrow 4; they are also strong. All other positive feedback loops are consistent and intersect \{1,22\}. That is, \{1,22\} intersects all consistent positive feedback loops, and for each positive feedback loop C that is inconsistent and strong, J contains k_C = 1. Hence, \{1,22\} dominates the wiring diagram of g. Therefore, Theorem 4 gives the better upper bound 2^S = 4 on the number of steady states of f.

Acknowledgement

The research was funded by NSF grants CMMI-0908201 and DMS-1062878.

References


Alan Veliz-Cuba
received his Ph.D. from Virginia Tech in 2010. He is currently a postdoc in the Department of Mathematics at the University of Nebraska-Lincoln. His interests include mathematical biology with an emphasis on using discrete and algebraic tools to systems biology.

Kristina Buschur
received her B.A. from Kenyon College in Gambier, Ohio with a double major in Mathematics and Molecular Biology. She is currently doing her graduate work through the Carnegie Mellon - University of Pittsburgh Joint Ph.D. Program in Computational Biology.

Rose Hamershock
has a B.S. in Mathematics from Muhlenberg College and is pursuing her M.A. in Biostatistics at Boston University. She is currently a research assistant for the Health and Disability Group in Boston University’s School of Public Health, with her main focus on financing for children with special health care needs. Prior to joining the Catalyst Center, she was a participant in the National Science Foundation-funded Research Education for Undergraduates at the Virginia Bioinformatics Institute at Virginia Tech. She also participated in the Summer Institute for Training in Biostatistics at Boston University.
Ariel Kniss graduated from Bucknell University in 2011 and is currently a Biomedical Engineering Ph.D. Student at Georgia Tech and Emory University. Her current research interests lie in computational modeling of immune cells.

Esther Wolff graduated from Columbia University with a BA in Mathematics and from Rensselaer Polytechnic Institute with an MS in Applied Mathematics. She currently works as research assistant at the Institute for Technology Assessment at Massachusetts General Hospital.

Laubenbacher has been a Professor at the Virginia Bioinformatics Institute and a Professor in the Department of Mathematics at Virginia Tech since 2001. He is also an Adjunct Professor in the Department of Cancer Biology at Wake Forest University in Winston-Salem (NC) and Affiliate Faculty in the Virginia Tech Wake Forest University School of Biomedical Engineering and Sciences. Prior to these appointments Dr. Laubenbacher was Professor of Mathematics at New Mexico State University. He has served as Visiting Faculty at Los Alamos National Laboratories, was a member of the Mathematical Science Research Institute at Berkeley in 1998, and was a Visiting Associate Professor at Cornell University in 1990 and 1993. Current interests in Dr. Laubenbacher’s research group include the development of mathematical algorithms and their application to problems in systems biology, in particular the modeling and simulation of molecular networks. An application area of particular interest is cancer systems biology, especially the role of iron metabolism in breast cancer.