A SAT-based Method for Counting All Singleton Attractors in Boolean Networks

Rei Higuchi[1] , **Takehide Soh**[1] , **Daniel Le Berre**[2] , **Morgan Magnin**[3] ,
Mutsunori Banbara[4] and **Naoyuki Tamura**[1]
[1]Kobe University,
[2]CRIL-CNRS UMR 8188, Université d'Artois,
[3]LS2N, UMR 6004,
[4]Nagoya University
{higuti@stu., soh@lion., tamura@}kobe-u.ac.jp, leberre@cril-lab.fr, morgan.magnin@ls2n.fr,
banbara@nagoya-u.jp

Abstract

Boolean networks (BNs) are widely used to model biological regulatory networks. Attractors here hold significant meaning as they represent long-term behaviors such as homeostasis and the results of cell differentiation. As such, computing attractors is of critical importance to guarantee the validity of a model or to assess its stability and robustness. However, this problem is quite challenging when it comes to large real-world models. To overcome the limits of state-of-the-art BDD- or ASP-based enumeration approaches, we introduce a SAT-based approach to compute fixed points (singleton attractors) of BN and exhibit its merits for counting singleton attractors of large-scale benchmarks well established in the literature.

1 Introduction

Boolean network (BN) [Kauffman, 1969b; Kauffman, 1993] is a mathematical model representing state transition systems. In a BN, each node is assigned a Boolean value of 0 (inactive) or 1 (active). The state of each node is then determined by a Boolean function based on the states of its input nodes in the previous discrete time step. The analysis of both static and dynamic properties of transition systems using BN has recently gained increasing importance in various fields. In particular, in biological systems, its application includes gene regulatory networks [Kauffman, 1969a; Daizhan Cheng, 2011], neural networks [Anthony, 2001], cell cycle control networks [Münzner *et al.*, 2019], and signal transduction networks [Mori *et al.*, 2015]. With the expanding range of such applications, repositories like `GINsim` [Naldi *et al.*, 2018], `CellCollective` [Helikar *et al.*, 2012] and `Biomodels` [Malik-Sheriff *et al.*, 2020], which describe biological networks in a formal format like BN, are being published and maintained.

Attractor is a steady state that a BN eventually reaches as time progresses. It can be either a *singleton attractor* (consisting of a single state) or a *periodic attractor* (cycling through multiple states). Attractors may also represent homeostasis, ensuring that the network can return to a steady state after a perturbation. Given its importance in understanding the system, this has led to extensive research on attractor detection [Mori and Akutsu, 2022]. In particular, analysis of singleton attractors (a.k.a. fixed points) plays a crucial role in understanding complex biological models since many biological systems stabilize to an observable phenotype rather than exhibit cyclic behavior. For instance, in gene regulatory networks, singleton attractors represent different cell types or states [Kauffman, 1993; Abou-Jaoudé *et al.*, 2016], thus helping understand how cells differentiate.

Attractor computation is thus an important subject. There are mainly two major approaches: BDD- and ASP-based methods [Garg *et al.*, 2008; Su and Pang, 2021; Klarner *et al.*, 2017; Benes *et al.*, 2022; Trinh *et al.*, 2023]. Those studies focused on enumerating attractors. However, recently published BNs are large and contain a huge number of attractors. For instance, the BN named "#124 NSP9-PROTEIN" consists of 252 nodes and has over 10^{20} singleton attractors. For such BNs, enumerating all attractors is neither realistic nor particularly useful. Indeed, there are BNs for which all attractors cannot be enumerated using state-of-the-art tools.

This limitation suggests *counting attractors* instead of enumerating them, as the number of attractors is still useful for assessing effects of intentional gene activation/inhibition or providing insights to network modelers. Thus, recent enumeration tools provide an option that returns only the number of attractors to reduce I/O costs, enabling retrieval of attractor count without displaying them. However, even with this feature, there remain BNs for which the number of attractors is still unknown. It is also challenging to thoroughly revise existing BDD- or ASP-based methods to adopt pure counting approaches. In the case of BDDs, a BDD is constructed as a compressed representation of all attractors, making enumeration and counting essentially equivalent. For ASP, it would be feasible if enumeration of stable models could be replaced with counting. However, the latest ASP encodings [Trinh *et al.*, 2023] include disjunctive rules, whereas the latest stable model counter [Kabir *et al.*, 2024] does not support disjunctive rules, making this approach difficult to implement.

Boolean Satisfiability Testing (SAT) problem [Biere *et al.*, 2021] is the problem of determining if there exists an assignment of variables that satisfies a given Boolean formula. SAT

has widespread applications in areas such as model checking, scheduling, automated planning, constraint satisfaction, and more [Biere *et al.*, 2021]. Due to its significance, SAT has garnered extensive research attention, leading to the development of various efficient SAT solvers, whose input is formulas in Conjunctive Normal Form (CNF). Advances in SAT technologies have extended their applicability to a range of challenging and practical combinatorial problems. Furthermore, SAT has several variants, such as AllSAT and #SAT whose goals are to enumerate or count all models of a given formula. In particular, for SAT solvers and #SAT solvers, there are international competitions every year[1], and these competitions witness the improvement of the performance of SAT and #SAT solvers.

In this paper, we propose a SAT-based method for counting all singleton attractors of a given BN. To fully carry the power of the state-of-the-art solvers, the difficulty is twofold: the Boolean formulation of singleton attractors such that models of the formula and attractors have one-to-one correspondences; the translation of arbitrary formulas into CNF formulas to allow the use of standard SAT or #SAT solvers. To tackle this, we develop a Boolean formulation of singleton attractors and study three CNF translation methods, including one using the classical Tseitin translation [Tseitin, 1968], one using a novel small implicant translation, and the last one using the hybridization of the other two approaches. We evaluated the performance of the proposed method with the state-of-the-art tools for BNs by comparing them with the 624 instances used in the literature [Trinh *et al.*, 2023] and 19 additional instances obtained from the BBM repository updates. The contributions of this paper are summarized as follows:

- A Boolean formulation (not in CNF) for singleton attractors in BNs whose models and attractors have one-to-one correspondence.
- The evaluation of three translation methods from this formulation to CNF formulas, including Tseitin, dedicated small implicants, and their hybridization.
- Comprehensive comparisons with state-of-the-art tools using 243 real-world and 400 artificial BNs.
- Determination of the number of singleton attractors for 9 real-world BNs and 73 large, crafted BNs, both of which have never been revealed before.

2 Preliminaries

2.1 Boolean Logic Recap

We start with the definition of a general form of *Boolean formulas*. Hereafter, unless otherwise specified, Boolean formula will be referred to as formula. A *Boolean variable* v is a formula. If P is a formula, $\neg P$ is also a formula. If P and Q are formulas, then $P \ \Delta \ Q$ are formulas, where Δ can be Boolean operators $\vee, \wedge, \leftrightarrow$, and $\rightarrow$. A disjunction of literals is called a *clause*. A conjunction of literals is called a *term*. A formula is in *Conjunctive Normal Form (CNF)* if it is a conjunction of one or more clauses. A formula is in *Disjunctive Normal Form (DNF)* if it is a disjunction of one or more terms. A formula is in *Negation Normal Form (NNF)* if it contains only the logical connectives $\wedge$, $\vee$, and $\neg$, and the $\neg$ operator applies only to literals. A *partial assignment* is a mapping from a subset of Boolean variables to the Boolean values 0 (false) or 1 (true). A *complete assignment* (or simply assignment) is a mapping from the set of Boolean variables to the Boolean values 0 (false) or 1 (true). We use the symbol α to denote assignments and extend it to formulas, defining a mapping that takes a formula and returns its truth value. A Boolean formula is *satisfiable* (SAT) if there exists an assignment that satisfies it and is *unsatisfiable* (UNSAT) otherwise. A *partial model* (or complete model) is a partial assignment (or complete assignment) that satisfies a given formula. A *SAT solver* is a program that computes a model (if any) from a CNF formula. A model counter/enumerator is a SAT solver specialized for counting/enumerating all models from a CNF formula.

2.2 Boolean Network and Attractor

Definition 1. A *Boolean network* (V, F) is defined as follows. $V = \{v_1, v_2, \ldots, v_n\}$ is a set of Boolean variables, also referred to as nodes. $F = (f_1, f_2, \ldots, f_n)$ is a vector of Boolean formulas, where f_i is called an *update function* for the node v_i. The update function $f_i : \{0,1\}^{k(i)} \rightarrow \{0,1\}$ is a Boolean formula over Boolean variables $V_i = \{v_{i,1}, v_{i,2}, \ldots, v_{i,k(i)}\} \subseteq V$, where k is a function that returns the number of variables contained in f_i. We call the set of variables V_i as the set of *input variables of* v_i.

When a BN represents a gene regulatory network, each v_i corresponds to each gene and values 0 and 1 correspond to the inactive and active states of genes. Suppose that the states of a BN evolve over discrete time points $\{0, 1, 2, \ldots\}$. Then, the *state* of a BN at a discrete time point $t \in \{0, 1, 2, \ldots\}$ is given by a vector $\vec{\boldsymbol{v}}(t) = (v_1^t, v_2^t, \ldots, v_n^t)$, where v_i^t represents the state of the node v_i at time t. The value of the node v_i at time $t+1$ is determined by the update function f_i applied to the values of its input nodes at time t as follows. Here we introduce f_i^t to simplify the description of f_i.

$$v_i^{t+1} = f_i(v_{i,1}^t, v_{i,2}^t, \ldots, v_{i,k(i)}^t) = f_i^t$$

Several update schemes were studied: synchronous (all nodes updated simultaneously), asynchronous (one node at a time), and generalized semantics (any subset of nodes updated simultaneously). These different semantics preserve singleton attractors [Paulevé and Sené, 2022]. As such, the discussion proceeds with asynchronous BNs without loss of generality. We define the relation of asynchronous transition T between two states $\vec{\boldsymbol{v}}(t)$ and $\vec{\boldsymbol{v}}(t+1)$ as follows.

$$T(\vec{\boldsymbol{v}}(t), \vec{\boldsymbol{v}}(t+1)) \overset{\text{def}}{\Longleftrightarrow}$$
$$\exists i \in [1, n].\, (v_i^{t+1} \leftrightarrow f_i^t) \wedge \bigwedge_{j=1, j\neq i}^{n} (v_j^{t+1} \leftrightarrow v_j^t)$$

This denotes that the value of at most one variable v_i can change between two states in the asynchronous transitions.

Definition 2. A *State Transition Graph (STG)* (S, T) of a BN (V, F) is defined as follows where $S = \{0,1\}^n$ is the

[1] https://satcompetition.github.io/, https://mccompetition.org/

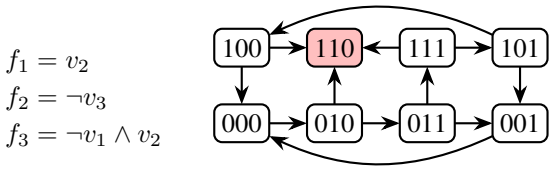

Figure 1: BN Example and its STG

set of states, and T is the transition relation given by the BN. The state transition graph is a directed graph consisting of the set of states S as nodes, and the arcs interpreted from the transition T.

Definition 3. A *Singleton Attractor* is a state that cannot transition to any other state but itself. In other words, a singleton attractor is a state that does not have outgoing arcs to any other node in STG.

Figure 1 illustrates a BN example and its STG, where `110` (red colored) is the only singleton attractor. Note that self-transition-loops are omitted in this Figure. Then, our target counting problem is defined as follows:

Definition 4. Given BN (V, F), a *Singleton Attractor Counting Problem (SACP)* is the problem of counting the number of all singleton attractors of (V, F).

3 Related Work and Existing Tools

There have been various tools that can compute attractors in BNs. Boolsim [Garg *et al.*, 2008] uses algorithms based on reduced ordered binary decision diagrams (ROBDD) and is capable of computing singleton and periodic attractors. Cabean [Su and Pang, 2021] is also a ROBDD-based method. PyBoolNet [Klarner *et al.*, 2017] uses the popular model checker `NuSMV` and ASP solvers. AEON [Benes *et al.*, 2022] uses ROBDD and is scalable to compute more than a trillion singleton attractors. fASP [Trinh *et al.*, 2023] uses an ASP-based method to enumerate singleton attractors.

Automata network (AN) [Folschette *et al.*, 2015] is another mathematical model for state transition systems and is a subclass of safe Petri nets [Paulevé *et al.*, 2012]. AN-ASP [Abdallah *et al.*, 2017] is an ASP-based method to calculate the attractors in AN. SAF [Soh *et al.*, 2023a] is a tool employing a SAT-based method [Soh *et al.*, 2023b] to calculate attractors in AN. While a BN can be represented as an AN via transformations using tools like BioLQM [Naldi, 2018], the transition rules in automata networks are different, and it is not possible to specify complex logical expressions in the firing conditions.

At the time of writing, no methods or tools focus solely on counting attractors, likely due to technical challenges (Section 1) and the adequacy of enumeration for medium-sized BNs. However, larger and more complex BNs have made enumeration insufficient, motivating our research. ASP-based methods, the state of the art for attractor enumeration, could support counting if a stable model counter were available. In 2024, sharpASP [Kabir *et al.*, 2024] was introduced, but its lack of support for disjunctive programs limits applicability to methods like fASP. If this is addressed, ASP-based methods could become strong candidates.

4 Proposed Methods

One approach to constructing our proposed SAT-based singleton attractor counter is to utilize existing SAT-based methods. As mentioned in the previous section, there is a SAT-based method for ANs [Soh *et al.*, 2023a], and by applying the BN-to-AN conversion tool BioLQM along with the latest model counter before and after this method, a SAT-based counter can be realized. However, preliminary experiments showed that this approach did not even outperform the best enumeration-based methods (see Section 5). Other approaches have applied Boolean formulas to AND/OR BNs (i.e., BNs consisting of AND/OR nodes), assuming that update functions are given in DNF. While they are theoretically capable of handling arbitrary functions, they do not provide concrete procedures for doing so [Tamura and Akutsu, 2009; Inoue, 2011]. In this section, we thus propose a constraint model for counting all singleton attractors in BNs with arbitrary update function and present methods to translate this model into CNF compactly.

4.1 Boolean Formulation of Singleton Attractors

We start with the Boolean formulation that must transition to any other state but itself. Suppose that v_i is the node whose value changes in the transition between t and $t+1$. If its value changes from 0 (at t) to 1 (at $t+1$) then $\neg v_i^t \wedge f_i^t$ holds at time t. Similarly, if its value changes from 1 (at t) to 0 (at $t+1$) then $v_i^t \wedge \neg f_i^t$ holds at time t. By negating those two constraints, we obtain the constraint for singleton attractors at time t as follows.

$$\Psi(t) = \bigwedge_{1 \le i \le n} \neg \left(\left(\neg v_i^t \wedge f_i^t \right) \vee \left(v_i^t \wedge \neg f_i^t \right) \right)$$

Since singleton attractors can be detected at any time point t, we arbitrarily consider the time $t = 0$. Thus, for simplicity, we omit the notation of ^{t}. Finally, we obtain the Boolean formula Ψ as follows.

$$\Psi = \bigwedge_{1 \le i \le n} (v_i \vee \neg f_i) \wedge (\neg v_i \vee f_i)$$

Proposition 1. Given BN (V, F), any complete model of the formula Ψ can be interpreted as a singleton attractor of (V, F) and vice versa.

Proof. ($\Rightarrow$) Let α be a model of Ψ consisting of a Boolean value vector $\vec{s} = (s_1, s_2, \dots s_n)$ such that $\alpha(v_i) = s_i$. It does not satisfy the two transition conditions $(\neg v_i \wedge f_i)$ and $(v_i \wedge \neg f_i)$. Then, the state $\vec{s}$ cannot transition to any other state but itself. Thus, $\vec{s}$ is an attractor of (V, F). ($\Leftarrow$) If a value vector $\vec{s}$ is an attractor of (V, F). Then, the state $\vec{s}$ does not transition to any other state but itself. A model α consisting of $\vec{s}$ does not satisfy the two transition conditions $(\neg v_i \wedge f_i)$ and $(v_i \wedge \neg f_i)$. Thus, α is a model of Ψ. □

To utilize the power of SAT techniques, we need to translate the arbitrary Boolean functions f_i into a CNF formula. In the following sections, we evaluate three methods to translate Ψ into a CNF formula.

4.2 Direct Translation

The most common way to translate an arbitrary Boolean function in CNF is to use the Tseitin translation [Tseitin, 1968]. Direct translation uses bidirectional Tseitin translation to convert the arbitrary formula Ψ into a CNF formula Ψ_D. It is performed by recursively replacing any "complex" subformula by a new variable and appending the definition of the new variable to the formula. The process terminates when the formula is either a conjunction or a disjunction of literals. Those definitions can be simply expressed by sets of clauses without adding new variables. We denote the Tseitin translation by TT. Using the Tseitin translation, Direct translation is given as follows.

$$\Psi_D = \bigwedge_{1 \leq i \leq n} (v_i \vee \neg l_i) \wedge (\neg v_i \vee l_i) \wedge TT(l_i \leftrightarrow f_i)$$

It is known that bidirectional Tseitin translation preserves the number of complete models, i.e. the original formula and the translated CNF have exactly the same number of complete models. Thus, the following proposition holds.

Proposition 2. Given BN (V, F), any complete model of the formula Ψ_D restricted to the original variables of Ψ is a complete model of Ψ and any complete model of Ψ can be extended to a complete model of Ψ_D.

Example 1. Suppose that $f_1 = (v_2 \vee v_3 \vee v_4) \wedge (v_2 \vee \neg v_5)$ is an update function of a given BN. By Direct translation described above, we obtain the following Boolean formula, where l_1, p_1 and p_2 are newly introduced auxiliary Boolean variables.

$$(v_1 \vee \neg l_1) \wedge (\neg v_1 \vee l_1)$$
$$(\neg p_1 \vee v_2 \vee v_3 \vee v_4) \wedge (\neg v_2 \vee p_1) \wedge (\neg v_3 \vee p_1) \wedge (\neg v_4 \vee p_1)$$
$$(\neg p_2 \vee v_2 \vee \neg v_5) \wedge (\neg v_2 \vee p_2) \wedge (v_5 \vee p_2)$$
$$(\neg l_1 \vee p_1) \wedge (\neg l_1 \vee p_2) \wedge (\neg p_1 \vee \neg p_2 \vee l_1)$$

This Direct translation can be implemented without any complex computation. However, the disadvantage of this translation is that sometimes the number of auxiliary variables and companion clauses becomes comparatively large, which can have a negative impact on the solver runtime as we will see in the experimental section.

4.3 Indirect Translation

In this approach, we will use again Tseitin traslation, but to feed the Boolean function to a model enumerator to obtain a DNF representation without adding new variables.

Let $\mathcal{M}$ be a function that translates a Boolean formula into the set of all its (complete) models. Then, for a formula ϕ, we can obtain a logically equivalent formula in disjunctive normal form (DNF) by using a DNF translation function DNF as follows.

$$DNF(\phi) \leftrightarrow \bigvee_{m \in \mathcal{M}(\phi)} \bigwedge_{l \in m} l$$

Then, using this translation, we can translate the singleton attractor formula Ψ into a CNF formula as follows, since the negation of a DNF is a CNF and the conjunction of two CNF is still a CNF:

$$\Psi_I = \bigwedge_{1 \leq i \leq n} \neg\left(DNF(\neg v_i \wedge f_i)\right) \wedge \neg\left(DNF(v_i \wedge \neg f_i)\right)$$

Then, the following proposition holds.

Proposition 3. Given BN (V, F), any complete model of formula Ψ_I can be interpreted as a complete model of Ψ and vice versa.

To perform this conversion using an off-the-shelf model enumerator, we use again the Tseitin translation. The models of $TT(f_i)$ contain both auxiliary variables from the Tseitin translation and the original variables of f_i. We can just filter out the auxiliary variables to get all the models of f_i from all the models of $TT(f_i)$.

However, the number of literals in each clause corresponds to the number of input variables for the function f_i, and the number of complete models may be exponential in the number of input variables of f_i. This can lead to Ψ_I with a large number of lengthy clauses, which may prevent the translation into DNF to complete or produce a formula too large for the SAT solvers or model counters. An alternative approach is to compute the prime implicants of the formula, i.e. subset-minimal sets of literals which satisfy the formula, instead of its models. We employ the translation-based method proposed in [Jabbour *et al.*, 2014]: each model of the translated CNF ψ_{PI} corresponds to a prime implicant of the original CNF ψ. Algorithm 1 shows the final procedure **PModelEnum**. We call **PModelEnum**(ψ, ∞) for implementing the aforementioned function $\mathcal{M}$.

The prime implicant of a Tseitin translated formula is not necessarily a prime implicant of the original formula. Take, for instance, the formula $f = (v_1 \wedge v_2) \vee (v_1 \wedge \neg v_2)$. It will be translated using the Tseitin translation as $TT(f) = (p_1 \vee p_2) \wedge (p_1 \leftrightarrow v_1 \wedge v_2) \wedge (p_2 \leftrightarrow v_1 \wedge \neg v_2)$. The single prime implicant of f is v_1. The prime implicants of $TT(f)$ are $p_1 \wedge \neg p_2 \wedge v_1 \wedge v_2$ and $\neg p1 \wedge p_2 \wedge v_1 \wedge \neg v_2$, which gives $v_1 \wedge v_2$ and $v_1 \wedge \neg v_2$ when restricted to the original variables, which is exactly the set of models, i.e. $\mathcal{M}(f)$. So computing the prime implicants of the Tseitin translated formula is a pragmatic effort to reduce the size of the DNF encoding. Note that an approach to compute exactly the PI of non clausal Boolean formula also exists [Previti *et al.*, 2015]. We implemented **PModelEnum** in Scala and use Sat4j [Le Berre and Parrain, 2010] as **ModelEnumerator** to perform all those calls to model enumerators in memory: for most benchmarks, the procedure takes less than 10 seconds.

Since we are interested in computing the prime implicants of our Boolean function f_i, we can simplify the Tseitin translation by only using an implication from the auxiliary variables to the original variables (instead of a bi-implication) and apply a negation normal form (NNF) [Darwiche and Marquis, 2002] to avoid negated auxiliary variables. This will reduce the number of clauses, not the number of variables, and will also affect the number of prime implicants found. Take for instance $f = (v_1 \wedge v_2) \vee (v_3 \wedge v_4)$, which has 2 prime implicants ($v_1 \wedge v_2$ and $v_3 \wedge v_4$). The one-direction Tseitin encoding will produce $1TT(f) = (p_1 \vee p_2) \wedge (p_1 \rightarrow v_1 \wedge v_2) \wedge (p_2 \rightarrow v_3 \wedge v_4)$

Algorithm 1 PModelEnum

Input: formula ϕ, cutoff co
Output: if enumeration succeeded or not, models M

```
Ω := one-direction Tseitin translation of NNF(φ)
Ω_PI := PI translation [Jabbour et al., 2014] of Ω
(isOK, M) := ModelEnumerator(Ω_PI, co)
M' := {decode(m) | m ∈ M}
return (isOK, M')
```

which has two prime implicants, $p_1 \wedge \neg p_2 \wedge v_1 \wedge v_2$ and $\neg p_1 \wedge p_2 \wedge v_3 \wedge v_4$ while $TT(f)$ has four prime implicants, $p_1 \wedge \neg p_2 \wedge v_1 \wedge v_2 \wedge \neg v_3$, $p_1 \wedge \neg p_2 \wedge v_1 \wedge v_2 \wedge \neg v_4$, $\neg p_1 \wedge p_2 \wedge \neg v_1 \wedge v_3 \wedge v_4$ and $\neg p_1 \wedge p_2 \wedge \neg v_2 \wedge v_3 \wedge v_4$.

Example 2. Suppose that $f_1 = (v_2 \vee v_3 \vee v_4) \wedge (v_2 \vee \neg v_5)$ is an update function. Using Algorithm 1, the following prime implicants of $\neg v_1 \wedge f_1$ and $v_1 \wedge \neg f_1$ are obtained:

$$\{\{\neg v_1, v_2\}, \{\neg v_1, v_3, \neg v_5\}, \{\neg v_1, v_4, \neg v_5\}\} \text{ and}$$
$$\{\{v_1, \neg v_2, \neg v_3, \neg v_4\}, \{v_1, \neg v_2, v_5\}\}, \text{ respectively.}$$

Then, we obtain the following CNF formula as Ψ_I.

$$(v_1 \vee \neg v_2) \wedge (v_1 \vee \neg v_3 \vee v_5) \wedge (v_1 \vee \neg v_4 \vee v_5)$$
$$(\neg v_1 \vee v_2 \vee v_3 \vee v_4) \wedge (\neg v_1 \vee v_2 \vee \neg v_5)$$

The upper three clauses are for $\neg v_1 \wedge f_1$, and the lower two clauses are for $v_1 \wedge \neg f_1$.

So an alternative way to obtain a DNF of ϕ is as follows:

$$DNF^P(\phi) \leftrightarrow \bigvee_{m \in \textbf{PModelEnum}(\phi, \infty)} \bigwedge_{l \in m} l$$

While not guaranteed that $DNF^P(\phi)$ represents the complete set of prime implicants, the impact of model compression is significant. For instance, we confirmed that for a BN rule, the number of clauses was reduced from 1,073,741,823 to 30 within a reasonable translation time, compared to the naive model-enumeration-based translation $DNF(\phi)$.

4.4 Proofs for Indirect translation

In this section, we will prove $DNF(\phi) \leftrightarrow DNF^P(\phi)$. To provide the proof, we start with the two properties of the Tseitin transformation in one direction. The full version of proof is available online [2].

Properties of One-direction Tseitin Translation. Let $A[B]$ be a source negation normal form (NNF) formula containing a subformula B. Note that an arbitrary formula can be converted to the NNF formula by using De Morgan's law. Let p be an auxiliary variable that does not appear in $A[B]$. Let A be an NNF formula such that A is obtained by all the appearances of B with the positive literal p. Then, a one-directional Tseitin translation of $A[B]$ for B can be performed by $A \wedge (\neg p \vee B)$. We clarify its properties with the following two propositions, which is based on the proof for equi-satisfiability in the literature [Tamura *et al.*, 2010].

Proposition 4. Suppose that α is a partial model of $A \wedge (\neg p \vee B)$, and α' is a partial model obtained by excluding p from α. Then $\alpha'(A[B]) = 1$ holds.

[2] https://doi.org/10.5281/zenodo.15462044

Proposition 5. Let α be a model such that $\alpha(A[B]) = 1$. Let α' be a model that extends α to the domain that includes p such that $\alpha'(p) = \alpha(B)$. Then $\alpha'(A \wedge (\neg p \vee B)) = 1$ holds.

Properties of PModelEnum. From here, we will also represent an assignment α as a set of literals for simplicity. For instance, suppose that ϕ is a formula over variables p, q, r. Then, an assignment $p = 1, q = 1, r = 0$ will be represented as $\{p, q, \neg r\}$. A partial assignment where $p = 1, q = 1$ will be represented as $\{p, q\}$. In addition, $\alpha(\phi) = 1$ means that any assignment satisfies $\bigwedge_{l \in \alpha} l$ also satisfies ϕ. The following proposition holds between $\mathcal{M}(\phi)$ and $\textbf{PModelEnum}(\phi, \infty)$.

Proposition 6. Given an arbitrary formula ϕ, for any $\alpha \in \mathcal{M}(\phi)$, there exists a partial model $\alpha' \in \textbf{PModelEnum}(\phi, \infty)$ such that $\alpha' \subseteq \alpha$ and $\alpha'(\phi) = 1$.

Proof. We prove it by contradiction. Assume that for some $\alpha \in \mathcal{M}(\phi)$, there does not exist a partial model $\alpha' \in \textbf{PModelEnum}(\phi, \infty)$ such that $\alpha' \subseteq \alpha$ and $\alpha'(\phi) = 1$. For such α, by Proposition 5, there exists a model β (which is α extended with auxiliary variables) that is a model of $1TT(\phi)$. Then, let β' be a prime implicant of $1TT(\phi)$. Obviously, $\beta' \subseteq \beta$ and $\beta'(1TT(\phi)) = 1$ hold. For such β', by Proposition 4, there exists a model α' that satisfies $\alpha'(\phi) = 1$. Clearly, $\alpha' \subseteq \alpha$ holds, which contradicts the assumption. Therefore, for any $\alpha \in \mathcal{M}(\phi)$, there exists a partial model $\alpha' \in \textbf{PModelEnum}(\phi, \infty)$ such that $\alpha' \subseteq \alpha$ and $\alpha'(\phi) = 1$. □

Then, we get the following proposition that gives the logical equivalence between $DNF(\phi)$ and $DNF^P(\phi)$.

Proposition 7. Given an arbitrary formula ϕ, any model of $DNF(\phi)$ is also a model of $DNF^P(\phi)$ and vice versa.

Proof. Let Φ_1 be $DNF(\phi)$ and Φ_2 be $DNF^P(\phi)$. ($\Rightarrow$) According to Proposition 6, for any $\alpha \in \mathcal{M}(\phi)$ there exists $\alpha' \in \textbf{PModelEnum}(\phi, \infty)$ such that $\alpha' \subseteq \alpha$. Thus, if Φ_1 has a satisfied disjunct, then Φ_2 also has a satisfied disjunct. Therefore, any model of Φ_1 is also a model of Φ_2. ($\Leftarrow$) According to Proposition 6, $\alpha'(\phi) = 1$. Therefore, there exists a complete model $\alpha \in \mathcal{M}(\phi)$, which is obtained by extending α' to a complete model. Therefore, any model of Φ_2 is also a model of Φ_1. □

4.5 Hybrid Translation

Direct translation could generate shorter clauses, but it needs many auxiliary variables, which explode the search space. Indirect translation does not need any auxiliary variables but could generate an exponential number of lengthy clauses. Therefore, we develop a hybrid method that combines the strengths of both encodings as follows:

$$\Psi_H = \bigwedge_{1 \le i \le n} \textbf{HybridTranslation}(v_i, f_i, co)$$

where **HybridTranslation** is detailed in Algorithm 2 and co is a cutoff value. The two translation methods are evaluated for each update function. The switching criterion is the number of models enumerated. If it becomes greater than co, Direct translation is used; otherwise, Indirect translation is used.

Algorithm 2 HybridTranslation

Input: v_i, f_i, co
Output: CNF formula
1: $(\text{isOK}^+, M^+) :=$ **PModelEnum**$(\neg v_i \wedge f_i, co)$
2: $(\text{isOK}^-, M^-) :=$ **PModelEnum**$(v_i \wedge \neg f_i, co)$
3: **if** $(\text{isOK}^+ \ \& \ \text{isOK}^-)$ **then return** $\neg(M^+ \cup \ M^-)$
4: **else return** $(v_i \vee \neg l_i) \wedge (\neg v_i \vee l_i) \wedge TT(l_i \leftrightarrow f_i)$

Since one of the two translation methods is used for each update function, the following proposition is obvious.

Proposition 8. Given BN (V, F), any complete model of the formula Ψ_H restricted to the original variables of Ψ is a complete model of Ψ and any complete model of Ψ can be extended to a complete model of Ψ_H.

Indirect translation generally performs better than Direct translation, but it requires model enumeration, which can be time-consuming. We explored various co values across some instances and found that 1000 strikes an effective balance, covering most BN rules within a reasonable translation time. Therefore, we used $co = 1000$ in our experiments.

5 Experiments

Environment. We conducted all experiments on a machine with Core i5 12400 (2.5GHz) of CPU and 64 GB of RAM. The time limit was 30 minutes per solver per benchmark.

Benchmark. We have used all the benchmarks employed in [Trinh *et al.*, 2023], which consist of a wide variety of Boolean networks. Biodivine Boolean Models (BBM) [Pastva *et al.*, 2023] include 230 networks sourced from all popular model repositories of Biological Systems such as `GINsim`, `CellCollective` and `Biomodels`, as well as individual publications. Pseudo-random (P-Random) consists of 400 random networks with 1000–5000 variables, preserving the structure of real-world networks, created in the literature [Trinh *et al.*, 2023]. This benchmark provides larger BN instances than the real-world networks. The selected BNs (Selected) are 13 instances of real-world BNs that were collected in [Trinh *et al.*, 2023].

Compared Solvers. As discussed in Section 3, since no attractor counter exists to the best of the authors' knowledge, we use the state-of-the-art enumeration tools for comparison: AEON [Benes *et al.*, 2022], fASP-c and fASP-s [Trinh *et al.*, 2023], PyBoolNet [Klarner *et al.*, 2017], and SAF [Soh *et al.*, 2023a]. The latter one uses BioLQM [Naldi, 2018] to translate each BN into AN. The runtime provided does include such translation time.

No Printing Options. To ensure the best option for counting, we configured all enumeration tools to not print attractors but only the number of attractors found, thereby avoiding I/O overhead. If we want to obtain all the solutions as a plain text file, the computational performance significantly slows down. For instance, AEON solved 224 instances without printing solutions within 1800 seconds, but it only solved 204 instances when printing solutions. This also indicates that the enumeration of real large BNs is impractical.

Proposed Method Setting. The three translation methods are implemented in Scala. We evaluated three state-of-the-art exact model counters from the 2023 competition, SharpSAT-TD [Korhonen and Järvisalo, 2023], d4 [Lagniez and Marquis, 2017], and GPMC [Ryosuke Suzuki and Sakai, 2017], on our SACP instances. SharpSAT-TD, the model-counting track winner, provided the best results. Additionally, we offer an enumeration solver using Hybrid translation and the All-SAT solver BDDMINISATALL [Toda and Soh, 2016]. We use D.C., I.C., and H.C. for Direct, Indirect, and Hybrid counting solvers, and H.E. for the Hybrid enumeration solver in the tables below. The runtime provided does include all translation times. Another proposed baseline-method is the one using the existing SAT-based method for ANs [Soh *et al.*, 2023a]. We configured SAF with BioLQM and SharpSAT-TD enabling to count singleton attractors of BNs (named SAF.C).

Consistency and Resource Availability. We have checked all numbers of attractors are consistent between all solvers. All benchmarks, solvers, and log files are available online[2].

Experimental results. Table 1 summarized the numbers of instances solved by each solver that are classified by A) benchmark series, B) range of the number of attractors $|A|$, and C) range of the number of nodes n. The first and second columns denote each category and the number of instances it contains. The remaining columns denote the number of instances solved by existing methods and proposed methods. In A), the proposed translation-based method was able to count the number of singleton attractors in almost half of the instances (between 313 and 330). In contrast, about one third of them could be enumerated by AEON (244), H.E. (240), and fASP-s (235). In B), the proposed method solved the largest number of instances over all ranges. In particular, the proposed method is the only tool capable of counting all singleton attractors when BNs contain more than 10^{30} of them. In C), the proposed method solved the largest number of instances over all ranges. In particular, the proposed method is the only tool capable of computing all singleton attractors in BNs whose number of nodes is greater than 2000. Note that AEON and fASP-s solved, respectively, 6 and 9 instances in that range, but all of those are UNSAT, which means there are no singleton attractors. Additionally, real-world instances have fewer than 500 nodes, except for two in Selected. All P-Random instances have more than 1000 nodes.

Table 2 lists BN instances where the number of singleton attractors was previously unknown and presents the computed values using the proposed approach. The first column denotes the network name and its source literature. The second and third columns denote the number of variables and the number of attractors identified in BN, respectively. This table demonstrates the ability of the proposed method to uncover previously unknown attractors in BN, and also highlights a challenging benchmark instance.

Impact of the SAT Translation on CNF Formulas. Table 3 shows the comparisons between the three proposed translation methods: Direct, Indirect and Hybrid. The first column denotes the initial of the benchmark set. The 2nd to 4th columns show the number of instances that cannot be translated within the time limit of 1800 seconds. The 5th column shows the average ratio of the number of variables between

		Existing Methods					Proposed Methods				
	#Ins.	PyB.	SAF	fASP-c	fASP-s	AEON	SAF.C	H.E.	D.C.	I.C.	H.C.
BBM	230	180	220	195	213	224	228	218	**230**	229	**230**
P-Random	400	0	0	15	15	12	0	15	71	**88**	**88**
Selected	13	5	6	6	7	8	10	7	**12**	11	**12**
Total	643	185	226	216	235	244	238	240	313	328	**330**
B) #Instances solved according to #Attractors $\|A\|$											
$0 \leq \|A\| < 1000$	157	138	142	**157**	**157**	154	142	**157**	**157**	**157**	**157**
$1000 \leq \|A\| < 10^{10}$	72	47	70	59	70	**72**	70	71	**72**	**72**	**72**
$10^{10} \leq \|A\| < 10^{30}$	27	0	14	0	8	18	24	12	**27**	25	**27**
$10^{30} \leq \|A\|$	387	0	0	0	0	0	2	0	57	**74**	**74**
C) #Instances solved according to #Variables n											
$0 \leq n < 100$	198	170	196	182	194	**198**	197	197	**198**	**198**	**198**
$100 \leq n < 1000$	44	15	30	19	26	34	41	28	**44**	42	**44**
$1000 \leq n < 2000$	111	0	0	6	6	6	0	6	41	**49**	**49**
$2000 \leq n$	290	0	0	9	9	6	0	9	30	**39**	**39**

Table 1: Number of instances solved by each solver

Instance		#Vars	#Attractors
#113 ER-STRESS	[Ostaszewski *et al.*, 2020]	182	116845500369426356109312 0
#122 NSP14	[Ostaszewski *et al.*, 2020]	168	33278627362665583108034953216
#124 NSP9-PROTEIN	[Ostaszewski *et al.*, 2020]	252	136112946768375385385349842972707284582 40
#144 SNF1-AMPK-PATHWAY	[Lubitz *et al.*, 2015]	202	10096027719780900754667077632
#220 H.-RESPONSE-IN-L.	[Ganguli *et al.*, 2015]	342	26563311466141754327040000
#221 MYCOBACTERIAL-L.	[Hegde *et al.*, 2012]	317	2473901162496
Cell-Cycle-Control	**[Romers and Krantz, 2017]**	**3158**	**—**
Alzheimer	[Aghamiri *et al.*, 2020]	762	135531809447440039244514002058631920 9- -710396035433027073714342803602931712 0
Cholocystokinin	[Aghamiri *et al.*, 2020]	383	47935169240579835005239296
Leishmania (same as #220 above) [Ganguli et al.*, 2015]*		*342*	*26563311466141754327040000*
Yeast-Pheromone	[Romers and Krantz, 2017]	246	5711631030629640192

Table 2: Number of singleton attractors computed by the proposed approach for real-world BNs on which it was unknown

Set	T.O.			#Variables		#Clauses		#Literals	
	D	I	H	I	H	I	H	I	H
B.	0	1	0	0.47	0.48	0.49	0.49	0.58	0.58
P.	0	0	0	0.22	0.22	0.52	0.52	1.52	1.52
S.	0	2	0	0.36	0.36	0.68	0.61	1.90	1.34

Table 3: Number of timeouts by each translation and average ratios between Direct and Indirect/Hybrid translations

Direct and Indirect translation. Similarly, the 6th column shows the average ratio of the number of variables between Direct and Hybrid translations. That is, if the figures are less than 1 then the number is smaller than the number of Direct translation. The following columns similarly denote the ratio of the number of clauses and the number of literals. We can see that Indirect translation cannot translate three instances within the time limit while Direct and Hybrid translation can translate all instances. We can also see that the number of variables and clauses of Indirect and Hybrid translation are smaller than the ones of Direct translation. However, the number of literals is larger than in Direct translation, except for BBM instances. The update functions in BBM instances often consist of a small number of variables but many operators such as $(v_1 \wedge (v_2 \vee v_3) \wedge \neg v_4) \vee (v_2 \wedge v_3 \wedge \neg v_4)$. In these cases, Direct translation generates more literals than Indirect or Hybrid translation.

6 Conclusion

In this work, we proposed a SAT-based method for counting singleton attractors in large Boolean networks (BNs). It determines the number of singleton attractors for 9 real-world and 73 crafted BNs that had never been revealed before, a result that holds some impact in the field of biological network research. Furthermore, the translation methods for converting arbitrary propositional formulas into compact CNF are considered applicable to other problems in the field of AI. As demonstrated by the results of the proposed baseline solver, our findings are not merely the outcome of applying a model counter but are instead a result of the careful refinements and techniques introduced in our translation approach. Future work is refinements on the algorithm, including leveraging existing logic minimizers, such as Espresso [3], to further reduce the DNFs obtained through Indirect translation.

[3] https://github.com/Gigantua/Espresso

Acknowledgements

This work was initiated under the collaborative research supported by the "PHC Sakura" program (43009SC, JPJSBP120193213), implemented by MESRI and JSPS, which ended in 2021. This work was also supported by JSPS KAKENHI (JP23K11047, JP24H00686, JP25K03097), and by ROIS NII Open Collaborative Research 2023 (23FP04).

References

[Abdallah *et al.*, 2017] Emna Ben Abdallah, Maxime Folschette, Olivier F. Roux, and Morgan Magnin. ASP-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks. *Algorithms for Molecular Biology*, 12(1):20:1–20:23, 2017.

[Abou-Jaoudé *et al.*, 2016] Wassim Abou-Jaoudé, Pauline Traynard, Pedro T. Monteiro, Julio Saez-Rodriguez, Tomáš Helikar, Denis Thieffry, and Claudine Chaouiya. Logical modeling and dynamical analysis of cellular networks. *Frontiers in Genetics*, 7, 2016.

[Aghamiri *et al.*, 2020] Sara Sadat Aghamiri, Vidisha Singh, Aurélien Naldi, Tomáš Helikar, Sylvain Soliman, and Anna Niarakis. Automated inference of Boolean models from molecular interaction maps using CaSQ. *Bioinformatics*, 36(16):4473–4482, 05 2020.

[Anthony, 2001] Martin Anthony. *Discrete Mathematics of Neural Networks*. Society for Industrial and Applied Mathematics, 2001.

[Benes *et al.*, 2022] Nikola Benes, Lubos Brim, Ondrej Huvar, Samuel Pastva, David Safránek, and Eva Smijáková. AEON.py: Python library for attractor analysis in asynchronous Boolean networks. *Bioinform.*, 38(21):4978–4980, 2022.

[Biere *et al.*, 2021] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[Daizhan Cheng, 2011] Zhiqiang Li Daizhan Cheng, Hongsheng Qi. *Analysis and Control of Boolean Networks—A Semi-tensor Product Approach*. Springer, 2011.

[Darwiche and Marquis, 2002] Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *J. Artif. Intell. Res.*, 17:229–264, 2002.

[Folschette *et al.*, 2015] Maxime Folschette, Loïc Paulevé, Morgan Magnin, and Olivier Roux. Sufficient conditions for reachability in automata networks with priorities. *Theoretical Computer Science*, 608:66–83, 2015.

[Ganguli *et al.*, 2015] Piyali Ganguli, Saikat Chowdhury, Shomeek Chowdhury, and Ram Rup Sarkar. Identification of th1/th2 regulatory switch to promote healing response during leishmaniasis: a computational approach. *Eurasip Journal on Bioinformatics and Systems Biology*, 2015(1):1–19, 2015.

[Garg *et al.*, 2008] Abhishek Garg, Alessandro Di Cara, Ioannis Xenarios, Luis Mendoza, and Giovanni De Micheli. Synchronous versus asynchronous modeling of gene regulatory networks. *Bioinform.*, 24(17):1917–1925, 2008.

[Hegde *et al.*, 2012] Shubhada R Hegde, Hannah Rajasingh, Chandrani Das, Sharmila S Mande, and Shekhar C Mande. Understanding communication signals during mycobacterial latency through predicted genome-wide protein interactions and boolean modeling. *PLoS One*, 7(3):e33893, 2012.

[Helikar *et al.*, 2012] Tomás Helikar, Bryan M. Kowal, Sean McClenathan, Mitchell Bruckner, Thaine W. Rowley, Alex Madrahimov, Benjamin Wicks, Manish Shrestha, Kahani Limbu, and Jim A. Rogers. The cell collective: Toward an open and collaborative approach to systems biology. *BMC Syst. Biol.*, 6:96, 2012.

[Inoue, 2011] Katsumi Inoue. Logic programming for boolean networks. In *In Proc. IJCAI-11*, pages 924–930, 2011.

[Jabbour *et al.*, 2014] Saïd Jabbour, João Marques-Silva, Lakhdar Sais, and Yakoub Salhi. Enumerating prime implicants of propositional formulae in conjunctive normal form. In *In Proc. JELIA2014*, volume 8761 of *LNCS*, pages 152–165. Springer, 2014.

[Kabir *et al.*, 2024] Mohimenul Kabir, Supratik Chakraborty, and Kuldeep S. Meel. Exact ASP counting with compact encodings. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *In Proc. AAAI-24*, pages 10571–10580. AAAI Press, 2024.

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

[Kauffman, 1969b] Stuart Kauffman. Homeostasis and differentiation in random genetic control networks. *Nature*, 224(5215):177–178, Oct 1969.

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

[Klarner *et al.*, 2017] Hannes Klarner, Adam Streck, and Heike Siebert. Pyboolnet: a python package for the generation, analysis and visualization of boolean networks. *Bioinform.*, 33(5):770–772, 2017.

[Korhonen and Järvisalo, 2023] Tuukka Korhonen and Matti Järvisalo. Sharpsat-td in model counting competitions 2021-2023, 2023.

[Lagniez and Marquis, 2017] Jean-Marie Lagniez and Pierre Marquis. An improved Decision-DNNF compiler. In Carles Sierra, editor, *In Proc. IJCAI-17*, pages 667–673. ijcai.org, 2017.

[Le Berre and Parrain, 2010] Daniel Le Berre and Anne Parrain. The Sat4j library, release 2.2. *Journal on Satisfiability, Boolean Modeling and Computation*, 7(2-3):59–64, 2010.

[Lubitz *et al.*, 2015] Timo Lubitz, Niek Welkenhuysen, Sviatlana Shashkova, Loubna Bendrioua, Stefan Hohmann, Edda Klipp, and Marcus Krantz. Network reconstruction and validation of the snf1/ampk pathway in baker's yeast based on a comprehensive literature review. *NPJ systems biology and applications*, 1(1):1–10, 2015.

[Malik-Sheriff *et al.*, 2020] Rahuman S. Malik-Sheriff, Mihai Glont, Tung V. N. Nguyen, Krishan K. Tiwari, Matthew G. Roberts, Ashley Xavier, Manh T. Vu, Jinghao Men, Matthieu Maire, Sarubini Kananathan, Emma L. Fairbanks, Johannes P. Meyer, Chinmay Arankalle, Thawfeek M. Varusai, Vincent Knight-Schrijver, Lu Li, Corina Dueñas-Roca, Gaurhari Dass, Sarah M. Keating, Young Mi Park, Nicola Buso, Nicolas Rodriguez, Michael Hucka, and Henning Hermjakob. Biomodels - 15 years of sharing computational models in life science. *Nucleic Acids Res.*, 48(Database-Issue):D407–D415, 2020.

[Mori and Akutsu, 2022] Tomoya Mori and Tatsuya Akutsu. Attractor detection and enumeration algorithms for boolean networks. *Computational and Structural Biotechnology Journal*, 20:2512–2520, 2022.

[Mori *et al.*, 2015] Tomoya Mori, Max Flöttmann, Marcus Krantz, Tatsuya Akutsu, and Edda Klipp. Stochastic simulation of boolean rxncon models: towards quantitative analysis of large signaling networks. *BMC Systems Biology*, 9(1):45, Aug 2015.

[Münzner *et al.*, 2019] Ulrike Münzner, Edda Klipp, and Marcus Krantz. A comprehensive, mechanistically detailed, and executable model of the cell division cycle in saccharomyces cerevisiae. *Nature Communications*, 10(1):1308, Mar 2019.

[Naldi *et al.*, 2018] Aurélien Naldi, Céline Hernandez, Wassim Abou-Jaoudé, Pedro T. Monteiro, Claudine Chaouiya, and Denis Thieffry. Logical modeling and analysis of cellular regulatory networks with ginsim 3.0. *Frontiers in Physiology*, 9, 2018.

[Naldi, 2018] Aurélien Naldi. Biolqm: a java toolkit for the manipulation and conversion of logical qualitative models of biological networks. *Frontiers in physiology*, 9:1605, 2018.

[Ostaszewski *et al.*, 2020] Marek Ostaszewski, Alexander Mazein, Marc E Gillespie, Inna Kuperstein, Anna Niarakis, Henning Hermjakob, Alexander R Pico, Egon L Willighagen, Chris T Evelo, Jan Hasenauer, et al. Covid-19 disease map, building a computational repository of sars-cov-2 virus-host interaction mechanisms. *Scientific data*, 7(1):1–4, 2020.

[Pastva *et al.*, 2023] Samuel Pastva, David Šafránek, Nikola Beneš, Luboš Brim, and Thomas Henzinger. Repository of logically consistent real-world boolean network models. *bioRxiv*, 2023.

[Paulevé and Sené, 2022] Loïc Paulevé and Sylvain Sené. Boolean networks and their dynamics: the impact of updates. *Systems Biology Modelling and Analysis: Formal Bioinformatics Methods and Tools*, pages 173–250, 2022.

[Paulevé *et al.*, 2012] Loïc Paulevé, Morgan Magnin, and Olivier Roux. From the Process Hitting to Petri Nets and Back. Technical Report hal-00744807, ETH Zürich, October 2012.

[Previti *et al.*, 2015] Alessandro Previti, Alexey Ignatiev, António Morgado, and João Marques-Silva. Prime compilation of non-clausal formulae. In Qiang Yang and Michael J. Wooldridge, editors, *In Proc. IJCAI-15*, pages 1980–1988. AAAI Press, 2015.

[Romers and Krantz, 2017] Jesper C. Romers and Marcus Krantz. rxncon 2.0: a language for executable molecular systems biology. *bioRxiv*, 2017.

[Ryosuke Suzuki and Sakai, 2017] Kenji Hashimoto Ryosuke Suzuki and Masahiko Sakai. Improvement of projected model-counting solver with component decomposition using sat solving in components (in japanese). In *JSAI Technical Report*, volume SIG-FPAI-103-B506, pages 31–36, 2017.

[Soh *et al.*, 2023a] Takehide Soh, Morgan Magnin, Daniel Le Berre, Mutsunori Banbara, and Naoyuki Tamura. SAF: SAT-based attractor finder in asynchronous automata networks. In *Proc. Computational Methods in Systems Biology (CMSB) 2023*, volume 14137 of *LNCS*, pages 175–183. Springer, 2023.

[Soh *et al.*, 2023b] Takehide Soh, Morgan Magnin, Daniel Le Berre, Mutsunori Banbara, and Naoyuki Tamura. Sat-based method for finding attractors in asynchronous multi-valued networks. In *In Proc. BIOSTEC 2023, Volume 3: BIOINFORMATICS*, pages 163–174. SCITEPRESS, 2023.

[Su and Pang, 2021] Cui Su and Jun Pang. CABEAN: a software for the control of asynchronous boolean networks. *Bioinform.*, 37(6):879–881, 2021.

[Tamura and Akutsu, 2009] Takeyuki Tamura and Tatsuya Akutsu. Detecting a singleton attractor in a boolean network utilizing SAT algorithms. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, 92-A(2):493–501, 2009.

[Tamura *et al.*, 2010] Naoyuki Tamura, Tomoya Tanjo, and Mutsunori Banbara. Constraint optimization problems and SAT encodings. *Journal of the Japanese Society for Artificial Intelligence*, 25(1):77–85, 2010.

[Toda and Soh, 2016] Takahisa Toda and Takehide Soh. Implementing efficient all solutions SAT solvers. *ACM J. Exp. Algorithmics*, 21(1):1.12:1–1.12:44, 2016.

[Trinh *et al.*, 2023] Van-Giang Trinh, Belaid Benhamou, and Sylvain Soliman. Efficient Enumeration of Fixed Points in Complex Boolean Networks Using Answer Set Programming. In Roland H. C. Yap, editor, *In Proc. CP 2023*, volume 280 of *LIPIcs*, pages 35:1–35:19, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Tseitin, 1968] Grigori S. Tseitin. On the complexity of derivations in the propositional calculus. *Studies in Mathematics and Mathematical Logic Part II*, pages 115—-125, 1968.