Trap spaces of multi-valued networks: definition, computation, and applications

Van-Giang Trinh [iD] [1],*, **Belaid Benhamou**[1], **Thomas Henzinger**[2], **Samuel Pastva** [iD] [2],*

[1]LIS, Aix-Marseille University, Marseille 13397, France
[2]Institute of Science and Technology, Klosterneuburg 3400, Austria

*Corresponding authors. LIS, Aix-Marseille University, Marseille, France. E-mail: trinh.van-giang@lis-lab.fr (V.-G.T.); Institute of Science and Technology, Klosterneuburg, Austria. E-mail: samuel.pastva@ist.ac.at (S.P.)

Abstract

Motivation: Boolean networks are simple but efficient mathematical formalism for modelling complex biological systems. However, having only two levels of activation is sometimes not enough to fully capture the dynamics of real-world biological systems. Hence, the need for multi-valued networks (MVNs), a generalization of Boolean networks. Despite the importance of MVNs for modelling biological systems, only limited progress has been made on developing theories, analysis methods, and tools that can support them. In particular, the recent use of trap spaces in Boolean networks made a great impact on the field of systems biology, but there has been no similar concept defined and studied for MVNs to date.

Results: In this work, we generalize the concept of trap spaces in Boolean networks to that in MVNs. We then develop the theory and the analysis methods for trap spaces in MVNs. In particular, we implement all proposed methods in a Python package called `trapmvn`. Not only showing the applicability of our approach via a realistic case study, we also evaluate the time efficiency of the method on a large collection of real-world models. The experimental results confirm the time efficiency, which we believe enables more accurate analysis on larger and more complex multi-valued models.

Availability and implementation: Source code and data are freely available at https://github.com/giang-trinh/trap-mvn.

1 Introduction

Boolean networks are simple but efficient mathematical formalism for modelling, analysing, and controlling complex biological systems (Schwab et al. 2020). Beyond systems biology, they have been widely applied in various areas from science to engineering (Schwab et al. 2020). Boolean network models of biological systems represent genes (or other species) as nodes that can take Boolean values: 1 (active) and 0 (inactive). However, having only two levels of activation may not be enough to fully capture the dynamics of real-world biological systems (Schaub et al. 2007). There are many examples (Schaub et al. 2007; Didier et al. 2011; Mushthofa et al. 2018) where the dynamics of the system can only be modelled by considering more than two activation levels. Hence, there is a crucial need to study multi-valued networks (MVNs), which are a generalization of Boolean networks (Naldi et al. 2007; Schaub et al. 2007).

1.1 Related work

Despite the importance of MVNs, only limited progress has been made on developing theories, analysis methods, and tools that can support them (Mushthofa et al. 2018). First, besides simulation, the analysis of logical models is mostly based around 'attractor' computation, since those correspond roughly to observable biological phenotypes (Schwab et al. 2020). For example, in gene regulatory and signalling networks, attractors can correspond to cell types, cell fates, and cyclic behaviour (e.g. circadian rhythms and cell cycles). Hence, analysis of attractors could provide new insights into

systems biology. However, finding all attractors of a logical model (even for the Boolean case) is challenging due to the complex dynamics of models (Schwab et al. 2020). The recent study of trap spaces of Boolean networks (Klarner et al. 2017) made a real breakthrough in the field of systems biology, as minimal trap spaces provide very good approximations of attractors and are much easier to compute. However, there has been no similar concept defined and studied for MVNs to date.

Furthermore, other biological properties, such as the gradual commitment of a cell to a specific phenotype, can be revealed through the lens of 'succession diagrams' constructed from the networks' 'maximal' trap spaces (Rozum et al. 2021, 2022). As such, minimal trap spaces are not the only relevant form of a trap space. In the Supplementary Data (Section S4.2), we give a more detailed discussion of trap space applications in biological modelling.

Second, most of the existing studies (see, e.g. Naldi et al. 2007; Schaub et al. 2007; Didier et al. 2011) focus on 'unitary' MVNs, with only very few studies focusing on 'general' MVNs (see, e.g. Mushthofa et al. 2018). Third, to the best of our knowledge, very few methods/tools [see, e.g. GINsim (Naldi et al. 2007), BMA (Benque et al. 2012)] have been developed for MVNs. Most analysis methods/tools for logical models are designed for Boolean networks only [see, e.g. GINsim (Naldi et al. 2007), PyBoolNet (Klarner et al. 2017), mpbn (Paulevé et al. 2020), and Trappist (Trinh et al. 2022)].

One notable issue is that the current supporting methods for MVNs cannot handle large and complex models (Naldi et al. 2007 ; Mushthofa et al. 2018). This issue also prevents

the modellers from building such models (Naldi et al. 2007), which could provide more accurate insights. Finally, a popular research direction is to convert an MVN to a Boolean network with similar dynamical behaviour, then applying the rich set of analysis methods/tools designed for Boolean networks. However, the existing Boolean encoding methods (e.g. the Van Ham Boolean mapping and its variants; Ham 1979; Didier et al. 2011) may not cover the full set of dynamics of the original MVN (Didier et al. 2011), and the encoding may even hinder the efficiency of the Boolean network methods/tools (Mushthofa et al. 2018). It is also worth noting that all mentioned encoding methods support only unitary MVNs. We believe that it is possible to develop direct and efficient methods for MVNs.

1.2 Our contributions

In this work, we study both general and unitary MVNs. Note that this inclusion of general networks is important not only in theory, but also in many biological applications (see Supplementary Section S1). First, we generalize the concept of trap spaces in Boolean networks to that in MVNs. Second, we prove several properties of trap spaces in MVNs including (i) the separation of minimal trap spaces, (ii) their relation with respect to the Van Ham Boolean mapping, and (iii) the characterization of trap spaces through Petri net (PN) siphons. Based on this characterization, we propose a new method utilizing answer set programming (ASP; Gebser et al. 2011) for computing different types of trap spaces of MVNs, including generic trap spaces, maximal trap spaces, minimal trap spaces, and fixed-points (a special sub-type of trap spaces). For fixed-points, we also consider another new method, which relies on the characterization of deadlocks of the PN encoding (Liu and Barkaoui 2016). We evaluate the method on a collection of real-world MVNs and show its applicability in treatment prediction on a case study of Myc-deregulation in breast cancer.

1.3 Paper outline

This article is structured as follows: In Section 2, we formalize the notion of trap spaces in MVNs and their relationship to the PN siphons. We propose a method based on answer-set programming for computing such trap spaces efficiently. In Section 3, we present a Python package `trapmvn` that implements this method for both SBML-qual and BMA models. We demonstrate the practical performance of the method on a wide selection of real-world models. We then use `trapmvn` as part of a case study to identify viable intervention targets in a model of Myc-associated deregulation in breast cancer. Finally, Section 4 discusses the scalability of the method, its future prospects, and highlights the role of trap spaces in reliable long-term behavioural analysis of logical models.

This article also has an associated supplement, in which we include the non-essential technical details of our methodology, as well as the full results of our benchmarks and case study. This article refers to this supplement where appropriate.

2 Methods

In this section, we define trap spaces of MVNs, discuss some of their theoretical properties, and finally present a method based on ASP for computing trap spaces of symbolically represented MVNs.

2.1 MVNs

Definition 1. *An MVN is a triple $\mathcal{M} = (V, K, F)$ such that:*

- *$V = \{v_1, \ldots, v_n\}$ is an indexed set of nodes (variables).*
- *$K = \{K_1, \ldots, K_n\}$ is an indexed set of integer intervals, representing the domains of variables $v_i \in V$.*
- *$F = \{f_1, \ldots, f_n\}$ is an indexed set of update functions of variables $v_i \in V$. Each f_i has a signature $f_i : \Pi_{j=1}^n K_j \to K_i$.*

With a slight abuse of notation, we write v_i to denote both the network node and the associated integer variable. Furthermore, we can also write K_{v_i} and f_{v_i} to denote the domain and update function of v_i, respectively. We define $x \in \prod_{j=1}^n K_j$ as the 'state' of the MVN, with x_i (or x_{v_i}) denoting the value of v_i in the state x. We can also write x as a vector $[x_1, \ldots, x_n]$. We then write $\mathcal{S}_{\mathcal{M}} = \prod_{j=1}^n K_j$ to denote the set of all states ('state space') of network $\mathcal{M}$. Finally, note that there are different possible formulations of update functions F, including fuzzy logic (Mushthofa et al. 2018), rule-based descriptions (Naldi et al. 2007; Delaplace and Ivanov 2020), or arithmetic expressions (Benque et al. 2012). We will return to this aspect when we discuss the encoding and manipulation of networks by our method.

The dynamics of an MVN are studied through its 'state-transition graph', $STG_{\mathcal{M}} = (\mathcal{S}_{\mathcal{M}}, \to)$. In particular, a 'trap' set of $STG_{\mathcal{M}}$ is a set of states that is closed with respect to $\to$. A trap set is called an 'attractor' if it is minimal, i.e. there is no other trap set that is a proper subset of this set. However, there are different ways of defining $STG_{\mathcal{M}}$, possibly leading to significantly different behavioural features.

First, we divide MVNs based on the variable update scheme as 'general' (Mushthofa et al. 2018) and 'unitary' (Schaub et al. 2007; Delaplace and Ivanov 2020). In the general networks, the variable changes follow the update functions exactly. Meanwhile, in the unitary networks, the value of v_i can only change by 'one level' at a time (i.e. $+1$ or -1). It has been argued that unitary networks better capture the continuity of biological interactions (Schaub et al. 2007), but is not true universally (Sun et al. 2014).

Note that the stepwise notion of unitary update can be in fact encoded directly into the update functions F. Specifically, we define $\hat{f}_v$ to represent the update function which adheres to the chosen variable update. That is, $\hat{f}_v(x) = f_v(x)$ for general MVNs and $\hat{f}_v(x) = x_v + d$ for unitary MVNs, where $d = 0$ if $x_v = f_v(x)$, $d = 1$ if $x_v < f_v(x)$, and $d = -1$ when $x_v > f_v(x)$.

The second aspect of MVN semantics is the concurrency of updates. Here, two prevalent approaches are the 'synchronous' (all variables update together) and 'asynchronous' update (exactly one variable updates non-deterministically in each step).

Note that for the same MVN, different update schemes may result in vastly different state-transition graphs (Thomas 1991). Consider the MVN $\mathcal{M}$ with $V = \{v_1, v_2\}$, $K_1 = \{0, 1\}$, and $K_2 = \{0, 1, 2\}$ (Example 2 in the Supplementary Data). Figure 1(a) shows the state transition
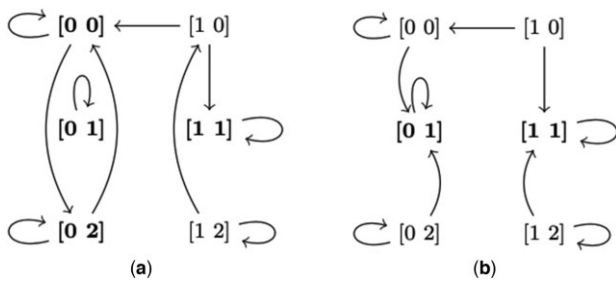
Figure 1. $STG_{\mathcal{M}}$ of the example MVN under general (a) and unitary update schemes (b). Attractors are highlighted in bold.

graph of $\mathcal{M}$ under the general update scheme. This graph has six trap sets: $\{[0\ 1]\}$, $\{[1\ 1]\}$, $\{[0\ 0], [0\ 2]\}$, $\{[0\ 1], [1\ 1]\}$, $\{[0\ 0], [0\ 1], [0\ 2]\}$, and $\{[0\ 0], [0\ 1], [0\ 2], [1\ 0], [1\ 1], [1\ 2]\}$. Out of these, three are attractors (minimal trap sets): $\{[0\ 1]\}$, $\{[1\ 1]\}$, and $\{[0\ 0], [0\ 2]\}$. Meanwhile, Fig. 1(b) shows the state transition graph of $\mathcal{M}$ using the unitary update scheme, which admits only two attractors: $\{[0\ 1]\}$ and $\{[1\ 1]\}$.

2.2 Trap spaces of MVNs

Overall, this variability under different update schemes motivates the study of 'trap spaces'. In the Boolean case, trap spaces have been shown to be a good approximation of network attractors, regardless of the chosen update concurrency (Paulevé et al. 2020). Here, we thus establish trap spaces for MVNs with a similar goal in mind:

> DEFINITION 2. *A space m of $\mathcal{S}_{\mathcal{M}}$ is a mapping which assigns each $v_i \in V$ a non-empty subset of K_i: $m(v_i) \subseteq K_i$. We write $\mathcal{S}_{\mathcal{M}}^{\star}$ to denote the set of all spaces of $\mathcal{M}$.*

With a slight abuse of notation, we can also interpret m as a subset of $\mathcal{S}_{\mathcal{M}}$, writing $x \in m$ for a state $x \in \mathcal{S}_{\mathcal{M}}$ when $x_v \in m(v)$ for all $v \in V$. Subsequently, we can define trap spaces as follows:

> DEFINITION 3. *A trap space is a space m such that for every state $x \in m$ and variable $v \in V$, we have $\hat{f}_v(x_v) \in m(v)$.*

In other words, in all states represented by m, all update functions only produce values that stay within m, meaning no $STG_{\mathcal{M}}$ (regardless of the update concurrency) can contain a transition which leaves m. Consequently, trap spaces of an MVN are independent on its concurrency update, is not true for the case of attractors. Note the use of $\hat{f}_v$, meaning that the notion of trap space respects the choice of unitary or general variable update.

When interpreting spaces $m \in \mathcal{S}_{\mathcal{M}}^{\star}$ as subsets of $\mathcal{S}_{\mathcal{M}}$, we can trivially establish a sub-space relation on $\mathcal{S}_{\mathcal{M}}^{\star}$, which allows us to reason about 'minimal' and 'maximal' trap spaces:

> DEFINITION 4. *A trap space m is minimal if and only if there is no trap space $m' \in \mathcal{S}_{\mathcal{M}}^{\star}$ s.t. $m' \subset m$. Analogously, a trap space m is maximal if and only if $m \subset \mathcal{S}_{\mathcal{M}}$, and there is no trap space $m' \in \mathcal{S}_{\mathcal{M}}^{\star}$ s.t. $m \subset m'$.*

Note that for the notion of maximal subspace, we require that $m \subset \mathcal{S}_{\mathcal{M}}$. Otherwise, a trivial space $m^{\star}$ which represents

all network states (i.e. $m^{\star}(v) = K_v$) is always a trap space, and always a superset of all other trap spaces.

For illustration, reconsider the example MVN (Fig. 1). Its general counterpart has six trap spaces as follows:

$$m_1 = \{0\}\{1\}, m_2 = \{1\}\{1\}, m_3 = \{0\}\{0, 2\},$$
$$m_4 = \{0, 1\}\{1\}, m_5 = \{0\}\{0, 1, 2\}, m_6 = \{0, 1\}\{0, 1, 2\}.$$

Out of these, three are minimal trap spaces (m_1, m_2, and m_3) and two are maximal trap spaces (m_4 and m_5). Meanwhile, its unitary counterpart has two minimal trap spaces ($\{0\}\{1\}$ and $\{1\}\{1\}$) and three maximal trap spaces ($\{0, 1\}\{0, 1\}$, $\{0, 1\}\{1, 2\}$, and $\{0\}\{0, 1, 2\}$). Finally, let us observe the following:

> THEOREM 1 (Separation of minimal trap spaces). *Let m_1 and m_2 be two distinct minimal trap spaces of $\mathcal{M}$. Then $m_1 \cap m_2 = \varnothing$.*

The proof is available in the Supplementary Data as Theorem 3.

By Definition 3 and the definition of a trap set, a trap space is also a trap set of an MVN for any chosen update concurrency. As a consequence, each trap space must contain at least one attractor of the MVN. By Theorem 1, any two distinct minimal trap spaces are disjoint; thus, the attractors contained in them are also disjoint. Hence, we can conclude that regardless of the chosen update concurrency, minimal trap spaces can be used as approximations of attractors in MVNs: The number of minimal trap spaces under-approximates the number of attractors (there can be other attractors that are not contained in minimal trap spaces), and each minimal trap space over-approximates at least a single attractor. Supplementary Section S3.3 presents additional examples of relations between attractors and minimal trap spaces.

2.3 Translation to Boolean networks

When $K_i = \{0, 1\}$ for all $v_i \in V$, we can refer to $\mathcal{M}$ as a 'Boolean network' $\mathcal{N} = (V, F)$ instead. For Boolean networks, the notion of trap spaces is well known, including efficient tools for computation of minimal and maximal trap spaces (Klarner et al. 2017; Paulevé et al. 2020; Trinh et al. 2022).

There are several works (Didier et al. 2011; Delaplace and Ivanov 2020) attempting to encode the multi-valued dynamics of an MVN into a Boolean network. Probably, the most widely used being the Van Ham encoding (Ham 1979), implemented in the tools GINsim (Chaouiya et al. 2011a, 2011b) and bioLQM (Chaouiya et al. 2013). The core idea of the Van Ham encoding is to expand each multi-valued variable $v \in V$ into $|K_v| - 1$ Boolean variables, such that an integer value $v = k$ is encoded as the truth value of the first k Boolean variables. The main advantage of this encoding is that any unitary state change only involves a single Boolean variable. The disadvantage is that the encoding admits invalid states which do not correctly encode any integer and may interfere with the actual dynamics of the model. More details about the Van Ham encoding are given in the Supplementary Section S2.3.

It is known that the Van Ham encoding preserves attractors of unitary networks (Didier et al. 2011). Nevertheless, it is unclear whether this mapping also preserves trap spaces in general. To answer this question, we present the following

proposition, the proofs for which can be found in the Supplementary Data.

PROPOSITION 1. *Let $\mathcal{N}_{\mathcal{M}}$ denote the Van Ham encoding of a MVN $\mathcal{M}$. Then, it holds that:*

- *There is a general MVN $\mathcal{M}$ for which neither the maximal nor minimal trap spaces of $\mathcal{N}_{\mathcal{M}}$ correspond to the respective trap spaces of $\mathcal{M}$.*
- *There is a unitary MVN $\mathcal{M}$ such that the maximal trap spaces of $\mathcal{N}_{\mathcal{M}}$ do not correspond to the maximal trap spaces of $\mathcal{M}$.*
- *For every unitary MVN $\mathcal{M}$, the minimal trap spaces of $\mathcal{N}_{\mathcal{M}}$ correspond to the minimal trap spaces of $\mathcal{M}$.*

This proposition states that the Van Ham encoding is still applicable for computation of minimal trap spaces of unitary networks, but cannot be used beyond this particular problem class.

2.4 Computing trap spaces through PN siphons

To tackle the problem of trap space computation of MVNs, we instead propose a different approach, based on an encoding of the problem into an answer-set programming query.

2.4.1 PN encoding of MVNs

First, let us establish a characterization of trap spaces through conflict-free PN siphons.

DEFINITION 5. *A one-safe PN is a bipartite-directed graph $\mathcal{P} = (P, T, W)$, where P and T are disjoint finite sets of vertices called places and transitions, respectively. Set W describes the arcs between places and transitions: $W \subseteq (P \times T) \cup (T \times P)$.*

A *marking* M of a one-safe PN is a subset $M \subseteq P$.

For $x \in P \cup T$, we write $\operatorname{pred}(x)$ and $\operatorname{succ}(x)$ to denote the predecessors and successors of x with respect to W (this notation naturally extends to subsets of $P \cup T$). The dynamics of a PN are dictated by the 'firing of transitions', such that a transition $t \in T$ can be fired non-deterministically in the marking M if $\operatorname{pred}(t) \subseteq M$. The result is a new marking $M' = (M \setminus \operatorname{pred}(t)) \cup \operatorname{succ}(t)$. This process defines a state-transition graph with vertices as the possible markings.

In the appendix of Chatain et al. (2014), the authors present a PN encoding of the 'asynchronous' MVN dynamics, which we briefly recall here. Note that this is not the only PN encoding of MVNs (see, e.g. Chaouiya et al. 2004, 2011a, 2011b), but to the best of our knowledge, the other encodings are not particularly suitable for the characterization of trap spaces.

Let $\mathcal{P}_{\mathcal{M}}$ denote the one-safe PN encoding of an MVN $\mathcal{M}$ based on Chatain et al. (2014). The places P of $\mathcal{P}_{\mathcal{M}}$ contain one place for every level of every MVN variable: $P = \cup_{v \in V} \{p_{v=i} | i \in K_v\}$. The set of transitions and arcs is then constructed such that the firing of a transition mirrors a possible update of an MVN variable. For example, let $u, v \in V$ be such that for every $x \in \mathcal{S}_{\mathcal{M}}$, we have $(x_v = k \wedge x_u = l) \Rightarrow \hat{f}_v(x) = o$ (with $o \neq k$). We then create a transition t with $\operatorname{pred}(t) = \{p_{v=k}, p_{u=l}\}$ and $\operatorname{succ}(t) = \{p_{v=o}, p_{u=l}\}$. Such transition 'moves' a token from $p_{v=k}$ to $p_{v=o}$ under the assumption

$u = l$. Note that there are many possible combinations of sets T and W that are valid for a particular $\mathcal{M}$. Later, we show how we compute these sets in our case.

Finally, let $m \in \mathcal{S}_{\mathcal{M}}^{\star}$ be a space of $\mathcal{M}$. We write that a set of places $M[m] \subseteq P$ is the 'mirror' of m when for every $v \in V$ and $i \in K_v$, $i \in m(v) \iff p_{v=i} \notin S$. That is, $M[m]$ contains the places corresponding to the 'inverse' of m. Observe that a state $x \in \mathcal{S}_{\mathcal{M}}$ is also a (trivial) space, and hence has a mirror $M[x]$.

2.4.2 Siphon characterization of MVN trap spaces

DEFINITION 6. *A PN siphon $S \subseteq P$ is a set of places such that for all $t \in T$, $(S \cap \operatorname{succ}(t)) \neq \varnothing$ implies $(S \cap \operatorname{pred}(t)) \neq \varnothing$.*

Intuitively, a *siphon S* preserves the condition that if all places in S are unmarked ($M \cap S = \varnothing$), the siphon remains unmarked. Furthermore, for a PN $\mathcal{P}_{\mathcal{M}}$, we say that a siphon is 'conflict-free' when for all $v \in V$, we have that $\{p_{v=i} | i \in K_v\} \setminus S \neq \varnothing$. That is, a conflict-free siphon must not contain 'all' places encoding a particular variable. Intuitively, every conflict-free siphon of $\mathcal{P}_{\mathcal{M}}$ represents a mirror of 'some' space $m \in \mathcal{S}_{\mathcal{M}}^{\star}$ (however, not all mirrors of spaces are siphons).

With this knowledge, we can observe the following theorem:

THEOREM 2. *Let $\mathcal{M}$ be an MVN and $\mathcal{P}_{\mathcal{M}}$ its PN encoding. Then a space $m \in \mathcal{S}_{\mathcal{M}}^{\star}$ is a trap space if and only if its mirror $M[m]$ is a conflict-free siphon of $\mathcal{P}_{\mathcal{M}}$.*

The proof is available in the Supplementary Data as Theorem 5.

Finally, note that we can define a partial order on siphons based on the subset relation, just as we did for trap spaces. Based on the definition of a mirror and the correspondence we just proved, we can easily deduce the following (see the proofs of Theorems 6 and 7 in the Supplementary Data):

PROPOSITION 2. *A trap space m of an MVN $\mathcal{M}$ is minimal if $M[m]$ is a maximal conflict-free siphon of $\mathcal{P}_{\mathcal{M}}$. A trap space m is maximal if $M[m]$ is a minimal conflict-free siphon.*

Observe that there are existing methods based on ASP (Trinh et al. 2022) and SAT solving (Nabli et al., 2016) that can handle enumeration of maximal (respectively, minimal) siphons and have been successfully used on Boolean networks before (Trinh et al. 2022). Our result shows that these methods should be also applicable to MVNs. Specifically, we aim to extend the ASP encoding from (Trinh et al. 2022) in order to support both maximal and minimal trap spaces of both general and unitary MVNs.

2.4.3 Siphon computation through ASP encoding

For now, we neglected the question of update function representation for MVNs, as this depends strongly on the chosen input format. In our work, we consider both SBML-qual (Chaouiya et al. 2013) and BMA (BioModelAnalyzer) (Benque et al. 2012) formats. These differ significantly in terms of both format and capabilities.

Ultimately, we represent an update function f_i using a series of 'binary decision diagrams' (BDDs) (Bryant 1986)

$B_1, \ldots, B_k$. The network variables are encoded into Boolean variables $\cup_{v \in V}\{p_{v=i}|i \in K_v\}$, just as in the case of the PN encoding. Subsequently, each BDD in the series gives the necessary and sufficient conditions for achieving a particular result level:

$$f_i(x) = \begin{cases} y_1 & B_1(x) = 1 \\ \ldots \\ y_k & B_k(x) = 1 \end{cases}$$

Here, $K_i = [y_1, \ldots, y_k]$ and B_i are individual BDDs, such that each BDD can be interpreted as a Boolean function on the encoded state x. We also require that for each state, there is exactly one BDD B_j s.t. $B_j(x) = 1$, hence the whole function is well defined. Also, note that such f_i can be easily transformed into $\hat{f}_i$, respecting either general or unitary variable update as desired.

This representation is relatively straightforward to obtain for SBML-qual models, as each update function is given as a list of Boolean terms over standard equality/inequality integer propositions. In this representation, the output values can repeat across multiple terms, and the function is not required to be exhaustive (there is a 'default' output value assigned to remaining inputs). However, this is easy to amend once each term is transformed into a symbolic BDD representation (see Supplementary Data).

For BMA, the translation process is more involved, as update functions are described through a language of algebraic expressions including addition ($+$), subtraction ($-$), multiplication ($\cdot$), and division ($/$), as well as other special functions like average and rounding. Furthermore, BMA employs a normalizing transformation on function inputs when the input domain differs from the output domain. That is, an input variable v in the range $K_v = [x, y]$ is normalized to the range $K_u = [a, b]$ when used in the update function f_u. As such, while BMA only admits integer variables, the update functions are more akin to rational functions.

While there are frameworks which partially support symbolic evaluation of such functions, e.g. algebraic decision diagrams (Bahar et al. 1997), we are not aware of any implementation that would support all the operations required by BMA. We thus opted to enumerate the whole function table and re-encode it back into individual BDDs.

Finally, to encode the dynamics of $\hat{f}$ into a PN, we consider all variable updates $(v = l) \rightarrow (v = k)$ (with k being $l \pm 1$ when $\mathcal{M}$ is unitary), and then enumerate all satisfying partial valuations w of the BDD $p_{v=l} \wedge \hat{B}_k$ ($\hat{B}_k$ being the BDD of $\hat{f}$ for output level k). For every valuation w, a transition $t \in T$ is created which moves the token from $p_{v=l}$ to $p_{v=k}$ while ensuring that for every other relevant $u \in V$, place $p_{u=w(u)}$ contains a token.

Note that the number of satisfying partial valuations w of a BDD (and transitively, the number of PN transitions) depends on the 'ordering' of Boolean variables within the BDD. However, computing the optimal variable ordering is a known non-trivial problem. To reduce the number of transitions in the resulting PN, we always test k randomized orderings for every update function and pick the one which produces the most compact PN.

Once the PN encoding $\mathcal{P}_{\mathcal{M}}$ is completed, the method proceeds based on the ASP encoding of the PN siphon problem proposed by Trinh et al. (2022). This encoding produces a query which can be processed by an ASP solver such as

`clingo` (Gebser et al. 2011), enumerating all maximal/minimal siphons of $\mathcal{P}_{\mathcal{M}}$. Technical aspects of this process are given in the Supplementary Data.

3 Results

We now present the results of our computational experiments. First, we establish the benchmark model dataset and describe the overall implementation of our tool `trapmvn`. We then present the performance evaluation of `trapmvn` in relation to the benchmark models and other tools (where applicable). Finally, we use the trap spaces arising under a wide range of therapeutic interventions in a Myc-deregulation model of breast cancer to assess the viability and reliability of such interventions.

3.1 Experiment setup

In our testing, we utilize SBML models from the BBM benchmark (https://github.com/sybila/biodivine-boolean-models/; note that BBM primarily publishes Booleanized SBML models, but the original multi-valued SBML files are also available.), which includes models from the GINsim tool (Chaouiya et al. 2011a, 2011b) repository as well as other independently sourced models. Furthermore, we include models from the BMA tool (http://biomodelanalyzer.org) repository (Benque et al. 2012). Disregarding trivial cases, we are left with 26 benchmark models. To the best of our knowledge, this is a highly representative sample of multi-valued models currently available in literature. The details of each benchmark model, technical description of the performance testing, as well as full results are given in Section 6 of the Supplementary Data.

3.2 The `trapmvn` package

We implement our method as a stand-alone open-source Python package `trapmvn`. The package provides a basic parser for both SBML and BMA models (JSON or XML). It implements the symbolic encoding for both formats using the BDD data structure from the `AEON.py` package (Beneš et al. 2022). From this symbolic encoding, we build either a 'general' or a 'unitary' one-safe PN. Such PN can be then encoded into the ASP query for the respective problem class and processed by the solver `clingo` (Gebser et al. 2011). All these steps are available to the user through several Python classes, but can be also handled directly by a simple command line interface. Finally, we also support export of the symbolic model representation back into SBML. This allows us to convert BMA models to SBML (to the best of our knowledge, this is not supported by any other tool).

3.3 Performance evaluation

Due to the lack of both theory and available tools, we are not aware of any existing study of trap spaces in MVNs. Nevertheless, as we have shown, the Van Ham encoding (Ham 1979) preserves the 'minimal' trap spaces under unitary semantics. Tools for computing minimal trap spaces of plain Boolean networks can be thus used in this case. We use `bioLQM` (Chaouiya et al. 2013) to compute the Van Ham encoding when necessary.

We can also consider fixed-points as a special (simpler) class of trap spaces. We can thus also compare with tools that specialize in this type of problem. Consequently, our
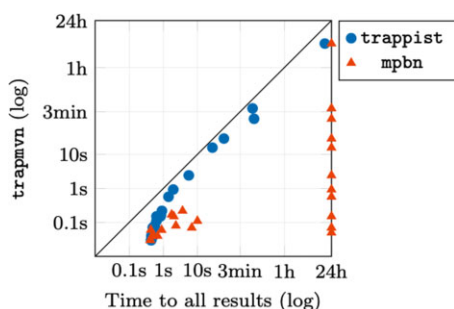
Figure 2. Relative performance of `trapmvn` compared with `trappist` and `mpbn` when counting all minimal trap spaces. The time scales are logarithmic. Points at the edge of the graph represent timeouts.

performance evaluation consists of three parts: 'minimal' and 'maximal' trap spaces, and 'fixed-points'.

3.3.1 Minimal trap spaces

To test the performance of `trapmvn` when computing minimal trap spaces, we compare its runtime to two state-of-the-art tools for trap space computation in Boolean networks, `trappist` (Trinh et al. 2022) and `mpbn` (Paulevé et al. 2020) (using the Van Ham encoding). Because the number of trap spaces can be large and the knowledge of all trap spaces is not always required, we consider two experiment settings: computing just 'one' and computing 'all' minimal trap spaces. The full results of this analysis are available in Supplementary Tables S5 and S6.

In general, `trapmvn` can handle all models in reasonable time for both the general and unitary semantics. Focusing on the unitary case, the difference in runtime when computing all minimal trap spaces is summarized in Fig. 2. Here, we see that `trapmvn` performs substantially better than `mpbn`, and is also always faster than `trappist`. However, the exact speed-up is hard to assess due to the presence of logarithmic time scales. As such, we also provide a simple box plot comparing the actual speed-up in Fig. 3. Here, we see that `trapmvn` outperforms both methods by a significant margin in both scenarios (i.e. first result and all results). We also compute the average speed-up weighted by the absolute runtime of each benchmark (i.e. longer running benchmarks are given higher weight), showing that when computing 'all' trap spaces, `trapmvn` is $3\times$ faster than `trappist`, and $58\times$ faster than `mpbn`.

3.3.2 Maximal trap spaces

The Van Ham encoding is not suitable for preserving 'maximal' trap spaces, and we are not aware of any existing method that can efficiently transform this problem to the domain of Boolean networks. As such, we cannot compare `trapmvn` to any other tool for this problem class. Nevertheless, we show that `trapmvn` can easily compute all maximal trap spaces of the considered benchmark models (Supplementary Table S7).

3.3.3 Fixed-points

Finally, for the case of fixed-points, there are two viable ASP encodings, one based on PN siphons and the other based on PN deadlocks. The encoding based on deadlocks is generally more efficient for this simpler problem class. As such, we implement both variants in `trapmvn` and compare the results to the Booleanized results from the tools `trappist` (Trinh et al.
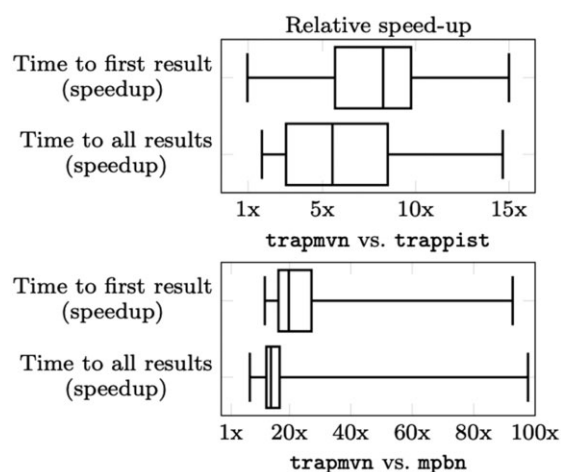

Figure 3. Relative speed-up in runtime of `trap-mvn` compared with `trappist` (top), res. `mpbn` (bottom).

2022) and `mpbn` (Paulevé et al. 2020), and multi-valued results from `AN-ASP` (Abdallah et al. 2017). Overall, our evaluation confirms the benefits of the deadlock-based encoding, showing a $2.6\times$ speed-up compared with the siphon encoding (Supplementary Table S8). Furthermore, we show that `trapmvn` also performs better than `trappist` or `mpbn` on fixed-point computation, but is outperformed by `AN-ASP` ($1.3\times$ speed-up), since `AN-ASP` is optimized solely for this specific type of problem.

3.4 Therapeutic interventions of Myc-deregulation

To demonstrate the practical utility of `trapmvn` in biological modelling, we present a case study expanding on the findings of Kreuzaler et al. (2019). The authors of Kreuzaler et al. (2019) use `BMA` (Benque et al. 2012) to explore viable therapeutic interventions in a large computational model of breast cancer through the lens of network attractors. Here, we show how to more reliably interpret the model by focusing on trap spaces instead.

3.4.1 Modelling Myc heterogeneity in breast cancer

The Myc transcription factor is one of the key coordinators in cell proliferation and regeneration (Kortlever et al. 2017). As such, oncogenic deregulations of Myc are commonplace in many cancers, breast cancer in particular (Vita and Henriksson 2006).

Still, most tumours have been shown to consist of several genetically distinct mutants, only some of which exhibit Myc overexpression (Gerlinger et al. 2012; Heselmeyer-Haddad et al. 2012). Such heterogeneity can impede some treatments, but it can also enable new therapies that target the cooperation between the mutants (Marusyk et al. 2014; Kreuzaler et al. 2019).

In the case of Myc-related mutations, an overexpression of Myc is linked to super-competitive behaviour that causes the cancerous cells to outproliferate their healthy neighbours. However, the same overexpression is also linked to greatly increased predisposition to apoptosis. In Kreuzaler et al. (2019), the authors reveal a mechanism by which Myc$^{\text{high}}$ mutants survive using a supply of Wnt1 transcription factor produced by a different, Myc$^{\text{low}}$ mutant.

This process is demonstrated both experimentally on *in vivo* mouse models, as well as *in silico* on a large-scale

multi-valued computational model. Based on the intervention response observed *in silico*, the authors identify a viable therapy targeting COX2 and MEK transcription factors and validate this therapy *in vivo*.

The model itself consists of 72 variables, ranging from 4 to 7 levels (`MYC IN-VIVO` in Supplementary Table S5). It is based on known literature, with additional validation and tuning using several independent datasets. There are five model variants: A healthy 'wild-type' (WT) model, Myc^{low} and Myc^{high} variants where the tumour consists homogeneously of a single mutant, and finally, mix-Myc^{low} and mix-Myc^{high} variants, which describe a heterogeneous tumour. In this case, the interaction is given as an outside assumption; there is no single model consisting of both Myc^{low} and Myc^{high} mutants sharing a state space.

To study the effects of possible therapeutic interventions, the authors compare the 'synchronous' attractors approximated by BMA across a range of single and dual variable knockouts. The effect of each intervention on the real-world phenotypes is assessed through variables `Apoptosis` and `Proliferation` which are directly embedded into the network. While this methodology is viable, it has shortcomings which we hope to address.

First, while BMA should be capable of computing the 'exact' synchronous attractors, the results in Kreuzaler et al. (2019) are only based on an 'approximate' method, due to the (lack of) scalability of the exact method. Second, the 'synchronous' update scheme can miss plausible model behaviour due to artificial synchronization between variables (Schwab et al. 2020). Meanwhile, trap spaces are universal regardless of the chosen update scheme (Paulevé et al. 2020). Finally, for non-trivial attractors, the case study in Kreuzaler et al. (2019) only considers the 'average' values of `Apoptosis` and `Proliferation`, which can be a poor approximation of the model's actual admissible behaviour. This is despite the fact that BMA 'can' also compute the estimated attractor intervals.

3.4.2 Single intervention effects

We start by replicating the single-variable knockout interventions performed in Kreuzaler et al. (2019), but in the context of trap spaces. Highlights from this analysis are presented in Table 1 (full data are available in the Supplementary Data). These results are in many aspects comparable to the original results obtained through BMA, however, they often paint a more complete picture of the model's behaviour.

First, there are four instances where our 'exact' method actually improves the precision of the original 'approximate' result (marked with an asterisk). Note that the more specific result is in all cases at the edge of the interval established by BMA, which means that it substantially differs from the 'average' considered in Kreuzaler et al. (2019).

Second, there are clearly many cases where the intervention causes the appearance of a non-trivial trap space, signified by an interval instead of a fixed value. Knowledge of these intervals is crucial when interpreting the effectiveness of interventions.

For example, consider the value of `Apoptosis` for the knockouts of Mcl1 and COX2 in mix-Myc^{low}. The average value is the same (i.e. 3), but the admissible interval is $[2, 4]$ for Mcl1 and $[0, 6]$ for COX2. The Mcl1 intervention guarantees `Apoptosis` ≥ 2, but we have no such expectation for COX2: Even though the 'best case' outcome is higher (i.e. $6 > 4$), the system is not guaranteed to visit these high-value states sufficiently often to trigger apoptosis.

3.4.3 Dual interventions with reliable and opportunistic effects

To systematically rank the high number of possible dual interventions, we propose to score each intervention I with two metrics, 'reliability' and 'opportunity', denoted by rel(I) and opp(I), respectively. The intuition is that these scores should represent the 'worst' and 'best' case scenarios (in terms of `Apoptosis` and `Proliferation`) admitted by the model for each intervention, regardless of the considered update

Table 1. Effects of selected single-node perturbations on proliferation (left) and apoptosis (right).

Intervention	WT	Myc^{low}	$\underset{Myc^{low}}{mix}$	$\underset{Myc^{high}}{mix}$	Myc^{high}
Akt	0	1...4	1...4	5	4
COX2	1	0...4	1...4	5	4
E2F-1	0	0	0	0	0
ErbB1	0	0...1	0	5	3
Erk	0	0...1	1*	5	3
Frizzled	0	0	0	3	3
Mcl1	1	4	4	5	4
Mek	0	0...1	0...1	5	3
mdm2	0	0...2	0...2	5	4
Noxa	1	4	4	5	4
p38	1	2...4	4	5	4
pRb	6	6	6	6	6
PTEN	1	4	4	6	5
Ras	0	0...1	0	5	3
VEGF	1	2	4	5	4

Intervention	WT	Myc^{low}	$\underset{Myc^{low}}{mix}$	$\underset{Myc^{high}}{mix}$	Myc^{high}
Akt	2	2...6	2...3	3	5
COX2	0	0...6	0...6	3	5
E2F-1	0	4	1...4	5	6
ErbB1	0	0...6	4	6	6
Erk	0	0...4	0*	6	6
Frizzled	0	0...6	3	5	5
Mcl1	1	4	2...4	4	5
Mek	0	0...4	0...4	6	6
mdm2	3	1...6	1...6	5	5
Noxa	0	3	0...3	3	3
p38	0	4	4	3	5
pRb	3	4*	1...4	3	5
PTEN	0	2	1	2*	4
Ras	0	0...6	4	6	6
VEGF	0	4	4	3	5

Intervals show non-trivial trap spaces. Values with asterisk show improvements over BMA approximate result.

Intervention (I)	Score		Apoptosis			Proliferation	
	$rel(I)$	$opp(I)$	WT	mix Myc^{low}	mix Myc^{high}	mix Myc^{low}	mix Myc^{high}
E2F-1 + Ras	10	10	0	4	6	0	0
ErbB1 + E2F-1	10	10	0	4	6	0	0
Beta-Catenin + E2F-1	9	9	0	3	6	0	0
Frizzled + E2F-1	9	9	0	3	6	0	0
p38 + E2F-1	9	9	0	4	5	0	0
VEGF + E2F-1	9	9	0	4	5	0	0
E2F-1 + Rsk	7	10	0	1...4	6	0	0
A1 + E2F-1	7	9	1	2...4	5	0	0
...							
VHL + Caspase3	-9	-9	0	0	0	4	5
VHL + Caspase9	-9	-9	0	0	0	4	5
PTEN + BAXBAK	-10	-10	0	0	0	4	6
PTEN + Caspase3	-10	-10	0	0	0	4	6
PTEN + Caspase9	-10	-10	0	0	0	4	6
BAXBAK + pRb	-12	-12	0	0	0	6	6
Caspase3 + pRb	-12	-12	0	0	0	6	6
Caspase9 + pRb	-12	-12	0	0	0	6	6

Figure 4. Eight best and worst dual interventions (out of 995 tested), together with their reliability and opportunity scores, as well as their effect on the relevant model variables.

concurrency. The exact definitions of these scores are given in Supplementary Section S5.4.

We then focus on the admissible dual-knockout interventions: both knockouts must be druggable (according to Kreuzaler et al. 2019), and their combination cannot increase Apoptosis beyond level three in the WT model. This leaves 995 interventions for which we compute both scores. For presentation purposes, we sort the interventions by the 'average' of these two scores.

The eight best and the eight worst interventions are shown in Fig. 4. As we can see, some of the best interventions admit a non-trivial trap space for the mix-Myc^{low} variant. However, even in this case, the reliability and opportunity scores are not vastly different. This raises a natural question regarding the prevalence of differences between reliability and opportunity scores in general. We further study this question in the Supplementary Data (Supplementary Section S5.4 and Supplementary Fig. S6, in particular), where we show that substantial differences between the two scores are in fact common even within the best scoring interventions.

4 Discussion

In this article, we formalized the notion of trap spaces in MVNs, then explored and proved properties of such trap spaces with applications in the analysis and control of MVNs. One notable property is that trap spaces of MVNs are independent to concurrency update schemes. We argued that, akin to Boolean networks, trap spaces can serve as approximation of network attractors. Furthermore, we showed that not all useful properties of trap spaces are preserved through a Boolean encoding of an MVN. For example, maximal trap spaces of unitary networks, crucial for the construction of the network's succession diagram (Rozum et al. 2021), are not preserved. Next, we made a connection between trap spaces of an MVN and siphons of its PN encoding. Based on this relationship, we proposed and implemented a new ASP method for computing different types of trap spaces of an MVN.

4.1 Method performance

We have evaluated the time efficiency of our method on real-world models collected from the literature. We show that the method scales well with the network size and it can handle large-scale realistic models for both the general and unitary semantics. The indirect approach (i.e. through a Boolean encoding) is only applicable for the case of fixed-points and minimal unitary trap spaces. In these cases, the direct approach (i.e. our method) outperforms the best indirect method.

In particular, we discuss the factors that contribute to the running time of each compared method (all are ASP-based) with respect to the minimal trap space computation. Aside from the absolute number of solutions, the practical complexity of an ASP query is affected by its number of atoms and its 'density', i.e. the ratio between the number of ASP rules and atoms. Through a systematic analysis (details are available in Supplementary Section S6.1), we see that an increase in density is always accompanied by increased runtime. Query density is clearly not the only indicator of ASP problem complexity. However, assuming we control for other relevant factors (solution count, tool/method, update scheme, model format, etc.), query density appears to be a relevant metric for comparing the complexity of two minimal trap space computation problems.

We also analyse the runtime of our method trap-mvn in the general and unitary cases. In general, trap-mvn needs more time for the general case than for the unitary case, as the PN encoding of a general MVN has more transitions than that of its unitary counterpart. The detailed discussions are given at the end of Section S6.1.

Finally, we have tried to also compare the performance of trap-mvn to BMA. However, in its current state, we were only able to run the 'approximate' attractor detection method, which is (as expected), much faster than any of the tested 'exact' methods. There appears to be no public documentation or reproducible artefact concerning the exact solver-based approach for the currently available version of BMA. However, the authors of Kreuzaler et al. (2019) note that the exact method was not able to efficiently analyse the Myc heterogeneity model. As such, it seems unlikely that it would provide competitive performance on the remaining models in our benchmark set.

4.2 Reliable identification of network interventions

Subsequently, we studied the practical applicability of our method on a model of Myc-deregulation from Kreuzaler et al. (2019). First, we indeed found instances where our exact method can improve the approximate results obtained in Kreuzaler et al. (2019). Furthermore, as we show in Table 1, the behaviour of the model can often admit significant fluctuations and uncertainty. These are not reflected in the average value within attractor states, as considered in Kreuzaler et al. (2019). This highlights the need for rigorous and exhaustive analysis of such behaviour uncertainty.

With this goal in mind, we propose a 'reliability' and 'opportunity' score to assess the viability of therapeutic interventions. Based on these scores, we see that even interventions that introduce some amount of uncertainty can be safely considered among the most viable (Fig. 4). Furthermore, we study how prevalent is the variability of intervention scores in our

dataset. As shown in Supplementary Fig. S6, for almost half of the interventions (439/995), the difference is zero. However, for 463/995 interventions, the model still admits a difference of three or more points. Such difference can impact the expected observed biological phenotypes if not taken into account. Furthermore, our results reveal that even though the highest score variability is associated with scores we would not consider particularly viable, very viable interventions (e.g. average score >5) can still exhibit high variability ($\leq$ 6).

Notably, the COX2 + MEK intervention chosen in Kreuzaler et al. (2019) has a reliability and opportunity score equal to 5, placing it on rows 65–149 of the table in Fig. 4 (there are 84 interventions with the same scores). While this is not among the best scores, it is still better than 85%–93% of interventions, meaning we retained the viability of this particular intervention. Furthermore, it should be noted that our screening for 'viable' interventions is rather rudimentary: in practice, some of the top perturbations might be ruled out due to factors other than the WT `Apoptosis` result (e.g. other side effects not captured by this model).

Overall, these results support the claim that logical models can exhibit a high level of variability in their outcomes and it is crucial to take this variability into account when drawing conclusions. Due to their favourable computational and theoretical properties, trap spaces appear to be uniquely suited for this task.

4.3 Applications and future prospects

Observe that the results we explored in this article can be also useful for other types of MVN analysis. First, there is a trap space-based model reduction technique. This relies on the fact that, given a single trap space, we can obtain a simpler MVN that captures the self-contained dynamics of the states within this trap space. Second, the set of minimal trap spaces of an MVN can be seen as an approximation of its attractors, regardless of the update concurrency. Third, there are existing control methods for Boolean networks based on trap spaces (Fontanals et al. 2020; Rozum et al. 2022). It should be possible to extend these methods to control of MVNs. We discuss the details of these applications in Section S3.3.

As a perspective, we also plan to attack the attractor detection problem for MVNs, as trap spaces only capture the network's static behaviour, whereas attractors can also capture its more complex dynamical aspects. Exploiting the relation between attractors and trap spaces of MVNs is a potentially promising direction for this problem. Furthermore, we plan to develop direct and efficient control methods for MVNs, because the control problem is crucial in systems biology. Arguably, it can be seen as 'a sequel' of the trap space or attractor analysis (Fontanals et al. 2020).

Supplementary data

Supplementary data are available at *Bioinformatics* online.

Conflict of interest

None declared.

Funding

Data availability

Source code and data are freely available at https://github.com/giang-trinh/trap-mvn.

References

Abdallah EB, Folschette M, Roux O *et al.* ASP-based method for the enumeration of attractors in non-deterministic synchronous and asynchronous multi-valued networks. *Algorithms Mol Biol* 2017;**12**: 20–3. https://doi.org/10.1186/s13015-017-0111-2.

Bahar RI, Frohm EA, Gaona CM *et al.* Algebric decision diagrams and their applications. *Formal Methods Syst. Des* 1997;**10**:171–206. https://doi.org/10.1023/A:1008699807402.

Beneš N, Brim L, Huvar O *et al.* AEON.py: python library for attractor analysis in asynchronous Boolean networks. *Bioinformatics* 2022; **38**:4978–80. https://doi.org/10.1093/bioinformatics/btac624.

Benque D, Bourton S, Cockerton C *et al.* BMA: visual tool for modeling and analyzing biological networks. In: Madhusudan P, Seshia SA. (eds.) *Computer Aided Verification*. Berlin, Germany: Springer, 2012, 686–692.

Bryant RE. Graph-based algorithms for boolean function manipulation. *IEEE Trans Comput* 1986;**C-35**:677–91. https://doi.org/10.1109/TC.1986.1676819.

Chaouiya C, Naldi A, Thieffry D. Logical modelling of gene regulatory networks with GINsim. In: van Helden J, Toussaint A, Thieffry D. (eds.) *Bacterial Molecular Networks*. New York, NY: Springer, 2011a, 463–79.

Chaouiya C, Remy E, Ruet P *et al.* Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets. In: Cortadella J, Reisig W. (eds.) *International Conference on Applications and Theory of Petri Nets*. Berlin, Germany: Springer, 2004, 137–56.

Chaouiya C, Bérenguier D, Keating SM *et al.* SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. *BMC Syst Biol* 2013;**7**:135. https://doi.org/10.1186/1752-0509-7-135.

Chaouiya C, Naldi A, Remy E *et al.* Petri net representation of multivalued logical regulatory graphs. *Nat Comput* 2011b;**10**:727–50. https://doi.org/10.1007/s11047-010-9178-0.

Chatain T, Haar S, Jezequel L *et al.* Characterization of reachable attractors using petri net unfoldings. In: Mendes P, Dada JO, Smallbone K. (eds.) *Computational Methods in Systems Biology*. Berlin, Germany: Springer, 2014, 129–42.

Delaplace F, Ivanov S. Bisimilar booleanization of multivalued networks. *Biosystems* 2020;**197**:104205. https://doi.org/10.1016/j.biosystems.2020.104205.

Didier G, Remy E, Chaouiya C *et al.* Mapping multivalued onto Boolean dynamics. *J Theor Biol* 2011;**270**:177–84. https://doi.org/10.1016/j.jtbi.2010.09.017.

Fontanals LC, Tonello E, Siebert H. Control strategy identification via trap spaces in Boolean networks. In: Abate A, Petrov T, Wolf V. (eds.) *Computational Methods in Systems Biology*. Berlin, Germany: Springer, 2020, 159–75.

Gebser M, Kaufmann B, Kaminski R *et al.* Potassco: the Potsdam answer set solving collection. *AI Commun* 2011;**24**:107–24. https://doi.org/10.3233/AIC-2011-0491.

Gerlinger M, Rowan AJ, Horswell S *et al.* Intratumor heterogeneity and branched evolution revealed by multiregion sequencing. *N Engl J Med* 2012;**366**:883–92. https://doi.org/10.1056/NEJMoa1113205.

Ham PV. How to deal with variables with more than two levels. In: Thomas R. (ed.) *Lecture Notes in Biomathematics*. Berlin, Germany: Springer, 1979, 326–43.

Heselmeyer-Haddad K, Berroa Garcia LY, Bradley A *et al.* Single-cell genetic analysis of ductal carcinoma in situ and invasive breast cancer

reveals enormous tumor heterogeneity yet conserved genomic imbalances and gain of myc during progression. *Am J Pathol* 2012;**181**: 1807–22. https://doi.org/10.1016/j.ajpath.2012.07.012.

Klarner H, Streck A, Siebert H *et al.* PyBoolNet: a python package for the generation, analysis and visualization of Boolean networks. *Bioinformatics* 2017;**33**:770–2. https://doi.org/10.1093/bioinformatics/btw682.

Kortlever RM, Sodir NM, Wilson CH *et al.* Myc cooperates with Ras by programming inflammation and immune suppression. *Cell* 2017; **171**:1301–15.e14. https://doi.org/10.1016/j.cell.2017.11.013.

Kreuzaler P, Clarke MA, Brown EJ *et al.* Heterogeneity of Myc expression in breast cancer exposes pharmacological vulnerabilities revealed through executable mechanistic modeling. *Proc Natl Acad Sci USA* 2019;**116**:22399–408. https://doi.org/10.1073/pnas.1903485116.

Liu G, Barkaoui K. A survey of siphons in petri nets. *Inf Sci* 2016;**363**: 198–220. https://doi.org/10.1016/j.ins.2015.08.037.

Marusyk A, Tabassum DP, Altrock PM *et al.* Non-cell-autonomous driving of tumour growth supports sub-clonal heterogeneity. *Nature* 2014;**514**:54–8. https://doi.org/10.1038/nature13556.

Mushthofa M, Schockaert S, Hung LH *et al.* Modeling multi-valued biological interaction networks using fuzzy answer set programming. *Fuzzy Sets Syst* 2018;**345**:63–82. https://doi.org/10.1016/j.fss.2018.01.003.

Nabli F, Martinez T, Fages F *et al.* On enumerating minimal siphons in Petri nets using CLP and SAT solvers: theoretical and practical complexity. *Constraints* 2016;**21**:251–76. https://doi.org/10.1007/s10601-015-9190-1.

Naldi A, Thieffry D, Chaouiya C. Decision diagrams for the representation and analysis of logical models of genetic networks. In: Calder M, Gilmore S. (eds.) *Computational Methods in Systems Biology.* Berlin, Germany: Springer, 2007, 233–47.

Paulevé L, Kolčák J, Chatain T *et al.* Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nat Commun* 2020; **11**:1–7. https://doi.org/10.1038/s41467-020-18112-5.

Rozum JC, Deritei D, Park KH *et al.* Pystablemotifs: python library for attractor identification and control in Boolean networks. *Bioinformatics* 2022;**38**:1465–6. https://doi.org/10.1093/bioinformatics/btab825.

Rozum JC, Gómez Tejeda Zañudo J, Gan X *et al.* Parity and time reversal elucidate both decision-making in empirical models and attractor scaling in critical Boolean networks. *Sci Adv* 2021;**7**:eabf8124. https://doi.org/10.1126/sciadv.abf8124.

Schaub MA, Henzinger TA, Fisher J. Qualitative networks: a symbolic approach to analyze biological signaling networks. *BMC Syst Biol* 2007;**1**:1–21. https://doi.org/10.1186/1752-0509-1-4.

Schwab JD, Kühlwein SD, Ikonomi N *et al.* Concepts in Boolean network modeling: what do they all mean? *Comput Struct Biotechnol J* 2020;**18**:571–82. https://doi.org/10.1016/j.csbj.2020.03.001.

Sun Z, Jin X, Albert R *et al.* Multi-level modeling of light-induced stomatal opening offers new insights into its regulation by drought. *PLoS Comput Biol* 2014;**10**:e1003930. https://doi.org/10.1371/journal.pcbi.1003930.

Thomas R. Regulatory networks seen as asynchronous automata: a logical description. *J Theor Biol* 1991;**153**:1–23. https://doi.org/10.1016/S0022-5193(05)80350-9.

Trinh V, Benhamou B, Hiraishi K *et al.* Minimal trap spaces of logical models are maximal siphons of their Petri net encoding. In: Petre I, Păun A. (eds.) *Computational Methods in Systems Biology.* Berlin, Germany: Springer, 2022, 158–76.

Vita M, Henriksson M. The Myc oncoprotein as a therapeutic target for human cancer. *Semin Cancer Biol* 2006;**16**:318–30. https://doi.org/10.1016/j.semcancer.2006.07.015.