Mapping the attractor landscape of Boolean networks

Van-Giang Trinh

trinh.van-giang@lis-lab.fr

Kyu Hyong Park

kjp5774@psu.edu

Samuel Pastva

xpastva@fi.muni.cz

Jordan C Rozum

jrozum@binghamton.edu

September 30, 2024

Abstract

Boolean networks are popular dynamical models of cellular processes in systems biology. Their attractors model phenotypes that arise from the interplay of key regulatory subcircuits. A succession diagram describes this interplay in a discrete analog of Waddington's epigenetic attractor landscape that allows for fast identification of attractors and attractor control strategies. We present a new approach to succession diagram construction for asynchronously updated Boolean networks, implemented in the biologist's Boolean attractor landscape mapper, `biobalm`, a Python 3 library. We compare the performance of `biobalm` to similar tools and find a substantial performance increase in succession diagram construction, attractor identification, and attractor control. To illustrate the utility of `biobalm`, we perform the most comprehensive comparative analysis to date of the succession diagram structure in experimentally-validated Boolean network models of cell processes and random ensembles. We find that random models (including critical Kauffman networks) have relatively small succession diagrams, indicating simple decision structures. In contrast, non-random models from the literature are enriched in extremely large succession diagrams, indicating an abundance of decision points in their dynamics and suggesting the presence of complex Waddington landscapes in nature.

1 Introduction

Biomolecular networks underpin cellular decisions and are essential in genotype to phenotype mapping. They represent the interactions between molecular entities within a cell, such as genes, proteins, and small molecules. Their kinetic parameters, however, are notoriously difficult to measure or estimate. Fortunately, living systems are often qualitatively robust to these parameters (see e.g. [1, 2, 3]), motivating widespread use of qualitative modeling in systems biology, with *Boolean networks* (BNs) being especially popular [4, 5, 6, 7, 8, 9]. First introduced in a gene regulatory context by Kauffman [10] as a means to study canalization (epigenetic robustness) and the emergence of phenotypic order, BNs consist of interlinked Boolean automata: each automaton's state (ON or OFF) is dynamically updated by the states of its linked automata according to a fixed update rule. This state evolves (either *synchronously* or *asynchronously*) in discrete time steps, eventually converging to one of several *attractors* (minimal sets of states from which no escape is possible). These attractors then typically correspond to phenotypes of interest. BNs can exhibit ordered, disordered, or critical perturbation responses, which reflects the robustness of their associated biological phenotypes [11, 12, 13, 14, 15, 16]. This has important basic science implications, but also biomedical significance: key *driver nodes* that disrupt undesired phenotypes represent potential drug targets, in some cases experimentally validated [17, 18, 19, 20].

One approach toward understanding phenotype robustness is through the self-sustaining configurations of small subnetworks in a BN, called *stable motifs*. Stable motifs correspond to *trap spaces* within network dynamics—hypercubes in the state space from which there is no escape [21, 22]. Stable motifs have analogs in more fine-grained, quantitative models, such as Hill kinetics ODEs [23, 24], which makes them particularly relevant in biological context. A *succession diagram* (SD), roughly analogous to Waddington's canalization landscape [25], is a directed acyclic graph that describes how these trap spaces nest within one another, indicating how entering one region of the phenotypic space is predicated on (or forbidden by) entering another [21, 9]. The leaf nodes of the SD are the *minimal trap spaces*, each of which contains at least one attractor. In some situations, these are guaranteed to be in one-to-one correspondence with the attractors of the system [26, 27]. Even when this correspondence does not hold, identifying minimal trap spaces gives enormous computational advantages when searching for the remaining "motif-avoidant" attractors (i.e., complex attractors residing outside of minimal trap spaces by avoiding the activation of a certain stable motif) [28, 29, 30].

The role of stable motifs in determining the attractor landscape highlights the deep significance of positive feedback loops in biomolecular networks that

was studied in detail by Thomas [31, 32, 33, 34] and later leveraged by others (e.g., [35, 36, 3]) to explore robustness in biomolecular networks. Indeed, the network structure alone provides tight constraints on the attractor dynamics of a biomolecular network and on its controllability properties, for example through its feedback vertex sets (sets of nodes that intersect all cycles) [37, 38, 39]. Recently, combining network structure with additional information about the dynamics has led to computationally efficient methods for attractor identification and control [40, 41, 42, 43, 44].

Identifying one BN attractor (resp. all attractors) is NP-hard (resp. #P-hard) because it contains N-SAT as a sub-problem [45]. Fortunately, biologically significant BNs are often sparse, which can be leveraged in attractor identification algorithms. Still, critical bottlenecks remain that render many biologically important networks intractable. Previously, each of the authors of this work have independently explored three approaches to overcoming these bottlenecks.

First, `pystablemotifs` [29] leveraged parity and time-reversal transformations to extend and accelerate the iterative succession diagram methodology of [46] and was used to construct the first-ever exact attractor repertoires for genome-scale BNs [47]. A limitation of `pystablemotifs` is its need to frequently compute Blake canonical forms (i.e., all prime implicants) for all update rules and their negations, limiting its use to very sparse networks where such computations are easy. Following `pystablemotifs`, `AEON.py` [48] was released, using binary decision diagrams along with transition guided reduction [49] to dramatically improve the efficiency of graph exploration in attractor identification. Finally, and most recently, `mts-nfvs` was released [30]. It uses an alternate scheme, implemented within the `trappist` library [50], to identify trap spaces via Petri-net encodings, which are easier to compute than the Blake canonical form. Furthermore, it leverages properties of negative feedback vertex sets (NFVS) to more efficiently search for motif-avoidant attractors [51, 40]. It uses minimal trap spaces and preprocessing heuristics to simplify or avoid reachability analysis in most cases. Still, non-minimal trap spaces and their nesting relationships further improve the method.

Each of these three methods is faster than the last. The algorithm we present here incorporates advantages from each, along with new insights about how to efficiently build SDs, resulting in `biobalm`. It uses the iterative SD approach of `pystablemotifs`, efficient rule representation and symbolic state-space searching from `AEON.py`, and the trap space identification method and NFVS approach of `mts-nfvs`. We demonstrate substantial speed improvement compared to these prior methods. Our method enables systematic exploration of motif-avoidant attractors in large BNs, exact attractor identification and control in previously intractable experimentally-supported BNs, and analysis of SD scaling in random and non-random BNs to provide insight into the emergence of canalization in biology.

2 Background

In this section, we briefly introduce key concepts in Boolean modeling and the notation that we use to describe them. A fully formal description of these concepts is available in Supplementary Text S1.

2.1 Boolean networks

An *asynchronous Boolean network* (ABN) of dimension n, denoted B, is a non-deterministic dynamical system. *States* of B are n-dimensional Boolean vectors $x \in \mathbb{B}^n$, with x_v denoting individual vector components. Each network variable v is assigned a Boolean *update function* $f_v : \mathbb{B}^n \to \mathbb{B}$ that governs its time evolution. At each discrete time step, the value of a non-deterministically selected variable is updated to match the output of its update function. When the variables are indexed, we may write f_i to refer to f_{v_i} and x_i to refer to x_{v_i}. An example ABN with $n = 4$ is shown in Figure 1a.

Network dynamics The dynamics of ABN B are encoded in a *state transition graph* $STG(B)$ whose nodes are the states $x \in \mathbb{B}^n$ of B. An edge $x \to y$ exists in $STG(B)$ if and only if B can update from state x to state y in one time-step (i.e., $f_v(x) = y_v$ for some v). The $STG(B)$ corresponding to the network from

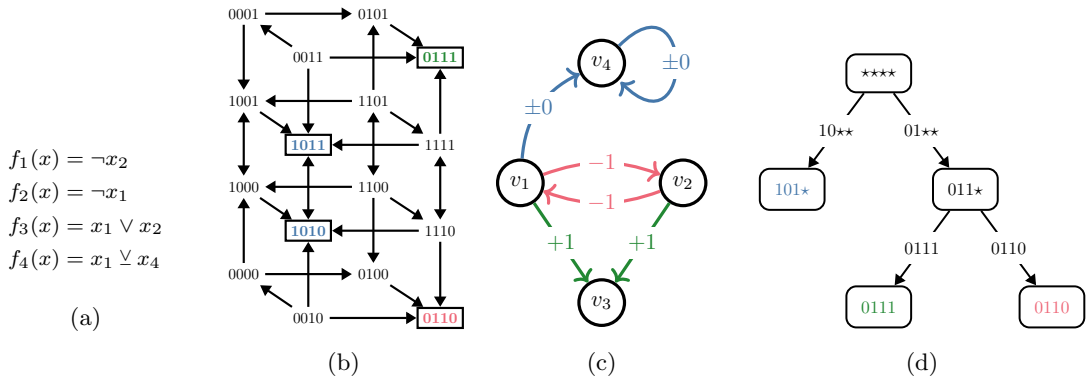


Figure 1: Illustration of the core concepts utilized in this paper. (a) A simple ABN $B = \{f_1, f_2, f_3, f_4\}$, with $\neg$ denoting negation, $\wedge$ conjunction, $\vee$ disjunction and $\veebar$ the exclusive or. (b) $STG(B)$. Self-loops are omitted for visual clarity. Attractor states are highlighted and distinguished by color. (c) $IG(B)$. Signs $+1$ (green), -1 (red), and ± 0 (blue) denote positive, negative, and non-monotonic influence, respectively. (d) $SD(B)$. Edges are labeled with the maximal trap spaces that percolate to the target nodes. As is typical (but not guaranteed), the minimal trap spaces (leaf nodes) and attractors (in (b)) coincide.

Figure 1a is shown in Figure 1b. The core feature of each $STG(B)$ are its *attractors*: minimal subsets of $\mathbb{B}^n$ that are closed under time evolution. These are also highlighted in Figure 1b. Note that other BN *update schemes* exist, such as the synchronous update that impact the dynamics (see [16] for a detailed discussion). However, much of the contributions of this paper do not depend on the chosen update scheme (further discussion is given in Supplementary Text S1.2).

Influence graph Boolean networks are so-named because their dynamics can be viewed as arising from a network of interactions among Boolean automata. These interactions are summarised by the *influence graph* (IG), denoted $IG(B)$, whose vertices (nodes) $v_1, \ldots, v_n$ are network automata. An edge from v_i to v_j indicates that the state x_i of automaton v_i is a non-redundant input [1] to the update function f_j. The sign of an edge from v_i to v_j can be -1 for inhibition, $+1$ for activation, or ± 0 if the impact of v_i depends on the remaining regulators. Figure 1c shows $IG(B)$ for the example network from Figure 1a.

Feedback vertex sets The IG is very useful for model analysis, as complex dynamics arise from the interplay between positive and negative feedback loops in the IG.[2] It is therefore often advantageous to identify a *feedback vertex set* (FVS), which is a set of nodes that intersects every cycle of $IG(B)$. Controlling the activation of an FVS is sufficient to drive an ABN into any of its attractors [37, 39]. Furthermore, a *negative* FVS (NFVS) is a set of nodes that intersects every negative feedback loop. Fixing the nodes of any NFVS ensures that all variables in an ABN eventually stabilize [51]. Typically, we are interested in an (N)FVS with as few nodes as possible. Identifying the true minimum (N)FVS is computationally difficult, but in our method, a heuristic estimate is

[1] In some applications, redundant edges added to $IG(B)$ may be informative, such as those that encode well-known interactions that are non-functional within the modeling context. Here however, we assume that $IG(B)$ only consists of essential interactions and is fully determined by B.

[2] Feedback loop sign is determined by the product of its edge signs. Cycles of sign 0 are treated as both positive and negative, because they may act as either, depending on where in the state space they are evaluated.

sufficient (computed using [48]). The IG in Figure 1c has two minimum FVSs: $\{v_1, v_4\}$ and $\{v_2, v_4\}$. Meanwhile, the only minimum NFVS is $\{v_4\}$.

2.2 Network trap spaces and succession diagrams

Network *subspaces* represent special subsets of $\mathbb{B}^n$ given by fixing some of the variables. Formally, subspaces are the members of $\mathbb{B}^n_\star$, where $\mathbb{B}_\star = \mathbb{B} \cup \{\star\}$. The value of a network variable v in a subspace $X \in \mathbb{B}^n_\star$ can be either fixed ($X_v = 0$ or $X_v = 1$), or free ($X_v = \star$). Each subspace X corresponds to a set of states $\mathcal{S}(X) \subseteq \mathbb{B}^n$ that agree with X in all fixed variables. For example, $X = 011\star$ corresponds to $\mathcal{S}(X) = \{0110, 0111\}$. We refer to X and $\mathcal{S}(X)$ interchangeably as context allows.

Trap spaces and attractors *Trap spaces* are subspaces that are closed under time evolution. Of special importance are (inclusion) *minimal trap spaces*, as each is guaranteed to contain at least one attractor. Importantly, however, attractors can also appear outside of the minimal trap spaces. For example, consider a simple ABN $f_1(x) = f_2(x) = x_1 \veebar x_2$. This network has two trap spaces, $\star\star$ and 00, of which 00 is minimal. However, the network also has two attractors: $A_1 = \{00\}$ and $A_2 = \{01, 10, 11\}$, meaning that A_2 does not lie within any minimal trap space. Such attractors are called *motif-avoidant* [42]. When a motif-avoidant attractor exists, the set of minimal trap spaces is called *incomplete* [22, 28]. Motif-avoidant attractors, as noted by [47, 16] and in Section 4.1, are rare, but it is very difficult to rule out their existence *a priori*. Various methods have been devised for detecting whether a set of minimal trap spaces is incomplete [28, 47, 40].

Subspace percolation The subspace *percolation* is the process of propagating fixed values among the network variables. We write $\mathcal{P}(X)$ to denote the one-step percolation of the subspace X. This updates each variable v which is free in X (i.e. $X_v = \star$) to a fixed value $b \in \mathbb{B}$ if and only if $f_v(x) = b$ for every state x in X. Repeatedly applying the $\mathcal{P}$ operator (up to n times) results in a subspace where no further variables can be updated, denoted $\mathcal{P}^\infty(X)$. We say that X *percolates* to the subspace $\mathcal{P}^\infty(X)$ and that X is *percolated* if $X = \mathcal{P}^\infty(X)$. For example, consider the ABN from Figure 1a and the subspace $X = \star 0 \star\star$. This one-step percolates to $\mathcal{P}(X) = 10\star\star$ because $f_1(x) = \neg x_2$, which is equivalent to $\neg 0$ for every network state in X (since $x_2 = 0$ for every $x \in X$). Meanwhile, for f_3 and f_4, both output values are possible in X, hence they remain free. A second application of $\mathcal{P}$ yields the subspace $101\star$ because $f_3(x) = x_1 \vee x_2$ is equivalent to $1 \vee 0$ for every state in $10\star\star$. Further applications of $\mathcal{P}$ result in no additional changes, because $f_4(x)$ simplifies to $\neg x_4$ in $101\star$. Therefore $\mathcal{P}^\infty(X) = \mathcal{P}^2(X)$, i.e. $\star 0 \star\star$ percolates to $101\star$.

Succession diagrams Knowledge of a network's trap spaces and their relationships with one another can aid in understanding the network's long-term dynamics, including its attractors [30, 29, 9] and response to interventions [21, 46, 47]. Percolated trap spaces are typically emphasized because any state in X eventually evolves to a state in the percolation of X. To formalize relationships between percolated trap spaces, [21] introduced *succession diagrams*. A succession diagram [3] of an ABN B, denoted $SD(B)$, is a rooted, directed acyclic graph. The vertices of $SD(B)$ are exactly all percolated trap spaces of B, with the edge relation describing how these nest within one another (by set inclusion). The root node is the percolation of $\star^n$.

Notice that the terminal (leaf) nodes of $SD(B)$ are exactly the minimal trap spaces of B. Furthermore, the successors of a node X correspond to the trap spaces obtained by percolating trap spaces that are subset-maximal within X. Borrowing terminology from related hypergraph structures [21, 47], we call such maximal trap spaces *stable motifs* and show them as edge labels when presenting succession diagrams. Most often, every maximal trap space percolates to a distinct succession diagram node. However,

[3] We use a novel, simplified definition of succession diagrams. Supplementary Text S1.3.3 gives a detailed comparison to the previous material on this topic.

in some cases, multiple maximal trap spaces percolate to the same subspace, in which case the edge can be annotated with multiple stable motifs. The succession diagram of the network from Figure 1a is depicted in Figure 1d.

Control interventions Succession diagrams are also useful for attractor control, and form the basis for several ABN control algorithms [46, 29]. The majority of these methods involve selecting a path in $SD(B)$ from the root node to a target trap space containing the desired attractor. At each branch point along the path, an intervention is selected to ensure that the system will eventually enter the trap space corresponding to the selected path. The union of these interventions drives the system to the target trap space with probability 1. The advantage of this approach is that it subdivides the attractor control problem into smaller, more manageable pieces. Typically, controlling entry into each trap space along the selected path involves fixing only a small subset of the Boolean variables. In `biobalm`, we have implemented two control algorithms from [29] with only slight modifications to allow for dynamic expansion of the succession diagram (see Supplementary Text S4 for details).

3 Algorithm

In this section, we provide a succinct description of the methods that are employed by the tool `biobalm`, the biologist's Boolean attractor landscape mapper, to construct and analyze succession diagrams. Visual summaries of key features and methods in `biobalm` are provided in Figures 2, 3, and 4. Further technical details (including algorithm pseudo-code and correctness proofs) are provided in Supplementary Text S2-5.

3.1 Succession diagram construction

We introduce several innovations to the algorithms of [21] and [47, 29]. At a high level, `biobalm` is broadly similar to the previous tools (Supplementary Text S2.3): the root node $\mathcal{P}^\infty(\star^n)$ of the ABN B is established by percolating the trivial trap space $\star^n$, and it is stored in a digraph $SD(B)$. Then, $SD(B)$ is further expanded by selecting a node (percolated trap space) X in $SD(B)$, identifying all maximal trap spaces within X, and percolating them to obtain the successor nodes. These are incorporated into $SD(B)$. Once the maximal trap spaces are computed for all nodes, the digraph $SD(B)$ is the succession diagram of B, as introduced in Section 2.

Compared to previous tools, however, `biobalm` has several key advantages. First, to compute the maximal trap spaces, we adapt the Petri-net-based approach to trap space identification of [40, 30]. Previous tools [29] employed the Blake canonical form (BCF, the disjunction of all prime implicants) for each function and its negation. Computing the BCFs can be a significant bottleneck, especially because they must often be recomputed for each SD node. Additionally, we improve the work of [40, 30] by proposing a new symbolic heuristic using binary decision diagrams (BDDs) that achieves a more compact Petri net encoding. We formulate rules for deriving this Petri net encoding for each SD node from the encoding of the full network, avoiding re-encoding the update functions percolated to each SD node (Supplementary Text S2.1).

The second major improvement is a more efficient implementation of the percolation operator $\mathcal{P}^\infty$ using a BDD representation (Supplementary Text S2.2). This implementation avoids the computationally expensive step of algebraically simplifying expression to obtain new BCFs after percolation.

Third, we adapt the attractor identification method of [30] to apply to arbitrary percolated trap spaces, greatly improving the efficiency of searching for motif-avoidant attractors (see Section 3.2 and Supplementary Text S3.3 for further details).

Finally, `biobalm` decouples the attractor identification from SD construction and can support *partially expanded* succession diagrams. This means that a large succession diagram need not be fully constructed for the attractor or control identification to run. This can provide partial (and under well defined conditions, complete) results for networks that have prohibitively complex succession diagrams. The remainder of this sub-section is dedicated to this partial

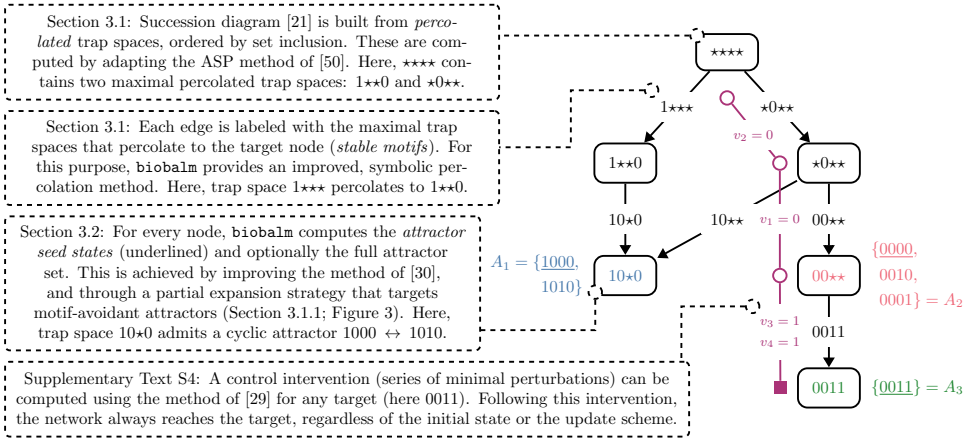


Figure 2: Summary of the `biobalm` features introduced in Section 3. The diagram is annotated with the network attractor sets and a chosen control intervention. The update functions and the STG of the network represented in this diagram are available in the Supplementary Text S8.1.

expansion strategy and its completeness conditions. We address the attractor identification in Section 3.2.

3.1.1 Partial expansion strategy

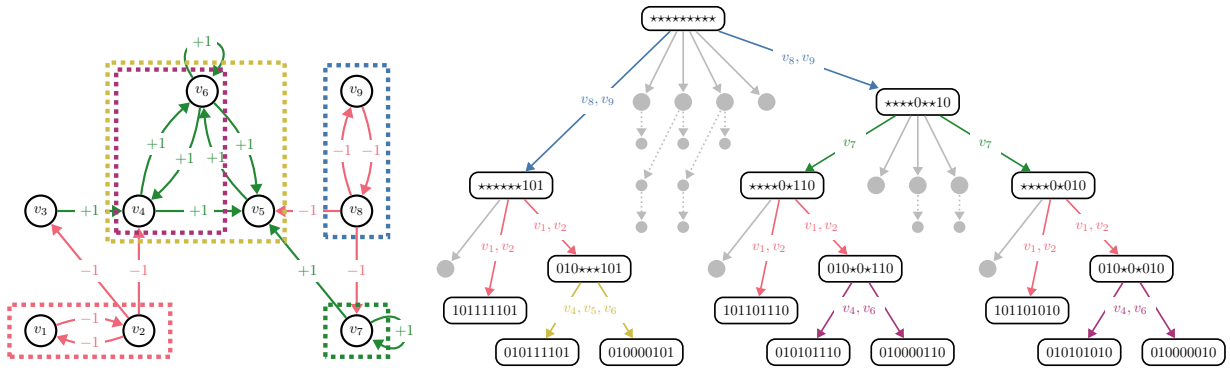
Previous methods [21, 29] construct $SD(B)$ by preferentially expanding deeper nodes. In `biobalm`, we implement multiple strategies, including depth-first and breadth-first expansion, allowing early termination if certain size or depth threshold is exceeded. We also developed several partial expansion strategies, which produce a sub-graph of $SD(B)$ in which certain nodes are not expanded, meaning their child nodes in $SD(B)$ are omitted. Upon completion, these methods produce partial SDs that are functionally equivalent to the full SD for attractor identification and control, but which eliminate computationally expensive (and cognitively burdensome) redundancies.

Of these strategies, the one that removes the most redundancies, and which we have selected as the default method for `biobalm`, is the *source block* expansion (Supplementary Text S5). Despite potentially eliminating a large portion of $SD(B)$, it nevertheless guarantees that for each attractor A of the network B, the smallest trap space that contains A, denoted $\mathcal{T}(A)$, is always found and expanded. Thus it properly identifies every leaf node (minimal trap space) of $SD(B)$, as well as every (non-leaf) node of $SD(B)$ containing a motif-avoidant attractor.

Here, we explain the principles of this method on a real-world example in Figure 3. This succession diagram represents the mammal sex-determination network developed in [52], with the values of the source nodes fixed to *true*, and certain redundant variables removed without modifying the SD structure (see also Supplementary Text S8.2).

Conceptually, our approach is similar to [53, 54] in that it identifies hierarchies of sub-networks (*blocks of variables*) that are mutually independent and can be processed separately. However, it combines this knowledge with the trap spaces to adapt these variable blocks to the context of each SD node. A simple variant of this principle is also present in `pystablemotifs` [29], which eliminates trap spaces arising from trivial combinations of source nodes (update functions of the form $f_i(x) = x_i$).

While expanding a non-minimal trap space X, we consider the network $B[X]$ that restricts the functions of B with the fixed values from X. The (inclusion) maximal trap spaces of $B[X]$ then correspond to the maximal trap spaces within X, which are normally used to compute the successors of X. For each such maximal trap space Y, we then define its *source*



(a) (b)

Figure 3: Example of a partial SD expansion by independent source blocks: (a) The IG of the simplified BN from [52] (see Supplementary Text S8.2 for details), with relevant variable blocks highlighted; (b) The partially expanded succession diagram. Colored edges are labeled with the expanded source blocks. Larger grey nodes represent trap spaces that are discovered but never expanded. Smaller grey nodes are never discovered, but appear in the full SD. For brevity, only one edge is shown per each non-expanded node.

block as the set of variables from which the fixed variables of Y are reachable in $IG(B[X])$. We then limit our consideration to inclusion-minimal source blocks. Often, such a source block is shared by multiple maximal trap spaces and is a top strongly connected component of $IG(B[X])$ (but can also be a larger set if the top components of $IG(B[X])$ admit no stable motifs).

We observe two key properties (proven in Supplementary Text S5): First, by expanding only stable motifs that share a single source block for each SD node, all minimal trap spaces remain reachable. This is because each source block is independent and can be expanded in any order; this partial expansion is equivalent to fixing an arbitrary preferred order. Second, if the sub-network induced by a source-block has no motif-avoidant attractors, then there is no motif-avoidant attractor in its corresponding SD node.

Together, these results motivate the following expansion procedure for a percolated trap space X: First, we identify maximal trap spaces within $B[X]$ and their source blocks, then arbitrarily select one source block C (we preferentially select small source blocks). We check the sub-network induced by C for motif-avoidant attractors. If it has none, we expand only the successors corresponding to C, and mark X as having no attractors. Otherwise, we must expand all successors of X in $SD(B)$ to determine which contain motif-avoidant attractors.

Returning to the example in Figure 3, in the root node of the succession diagram, there are six maximal trap spaces divided among four source blocks: $C_1 = \{v_8, v_9\}$, $C_2 = \{v_1, v_2\}$, $C_3 = \{v_7, v_8, v_9\}$, and $C_4 = \{v_1, \ldots, v_9\}$. Initially, we choose C_1 (blue), which in one branch also eliminates C_3. In the other branch, we then pick C_3 (green), which is now reduced to just v_7 (v_8 and v_9 are fixed). Next, we expand C_2 (red). This either leads directly into a minimal trap space, or it simplifies C_4 to just two (purple) or three (yellow) variables, depending on the choice in C_1. Notice that we expanded much fewer SD nodes (17) compared to the full SD (36). Also, the largest network ever considered in the attractor identification has only three variables instead of nine.

3.2 Attractor identification

In `biobalm`, we consider two variants of the attractor identification problem: First, the *attractor sets* problem is to determine every set of states that represents

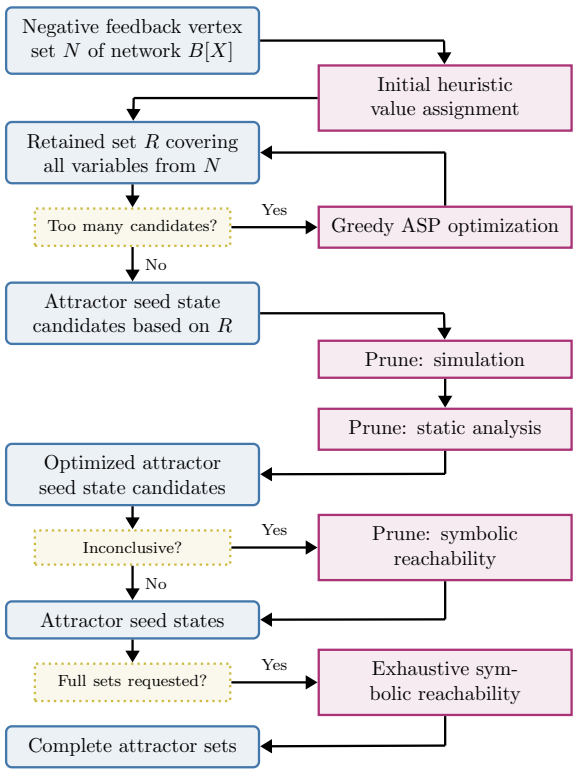


Figure 4: Attractor identification workflow for a fixed node (trap space) X of the succession diagram $SD(B)$. Blue and purple boxes represent data and computational steps, respectively.

an attractor of network B. This is how the attractor identification is typically understood, but it means that each attractor set must be sufficiently small such that it is fully identifiable. As this is not always the case for large complex attractors, we also consider the *attractor seed* problem, which is to identify exactly one representative *seed* state for each attractor. This allows us, for example, to identify the presence of motif-avoidant attractors without fully enumerating their states.

Our approach in `biobalm` is based on the method of `mts-nfvs` [30], which we have significantly improved and extended to apply to arbitrary percolated trap spaces rather than only the minimal ones. Our workflow applies to each SD node X individually. A summary is given in Figure 4 as well as Supplementary Text S3.

Recall that attractors cannot cross trap space boundaries, and that the $\mathcal{T}(A)$ (smallest trap space that contains A) for each attractor A is always a fully percolated trap space. Thus it is sufficient to search for attractors only within percolated trap spaces, i.e., the nodes of $SD(B)$. Furthermore, when searching an SD node X for attractors, we can disregard not only the states outside X, but also the states in each successor node Y (as these are considered separately when searching Y).

Following [30], we identify an NFVS U^- of $IG(B[X])$ (where $B[X]$ is the restriction of B to the subspace X) and assign a Boolean value to each variable of U^- to form the so-called *retained set*. Any variable assignment is valid, but we use heuristics to try to reduce later computations (see below and Supplementary Text S3.3). We create a modified STG by removing transitions that violate the fixed values prescribed by U^-. For example, if we assign $v \in U^-$ the value 0, we remove every transition that changes the value of variable v from 0 to 1 (transitions going from 1 to 0 are *retained*). We then identify the set of fixed points C of this modified STG, called a *candidate set*. It has been previously proven that C contains a viable seed state for every attractor in $B[X]$ (and thus in the trap space X); but C can also contain spurious candidates [51]. Crucially, using answer set programming (ASP) methods, it is possible to construct C without building the full STG of $B[X]$ or its modified coun-

terpart [51]. However, in some instances, the C can be too large to enumerate fully. We thus propose a novel ASP-based optimization strategy that dynamically adjusts the retained set subspace to produce a smaller set of candidate states (Supplementary Text S3.3).

Once candidates are fully identified, we iteratively eliminate redundant candidate states–those that can reach a stable motif or a member of C that has not yet been eliminated–through randomized simulation and static analysis (using the tool `pint` [55]). We meaningfully updated the original simulation algorithm from [30], achieving higher performance. Finally, if these pruning steps remain inconclusive, we adapt an exhaustive symbolic reachability method using `AEON.py` [48] that replaces an explicit model checking step proposed by [30]. The main advantage is that the symbolic representation enables exhaustive search of much larger STGs. A byproduct of exhaustive symbolic reachability is the full attractor sets. Nevertheless, if the knowledge of attractor seed states is sufficient, the symbolic reachability is rarely used due to the efficiency of the previous pruning methods.

4 Results

4.1 Benchmarks

To evaluate the overall effectiveness of `biobalm`, we consider the attractor seed identification problem over a large collection of real-world (230 networks with 14010 input configurations; from the BBM dataset [56]) and synthetic (2760 networks; critical N-K, nested canalyzing, and dense ensembles) Boolean networks. A detailed description of experiment setup is given in the Supplementary Text S6. We compare `biobalm` with `AEON.py` [48] and `mts-nfvs` [30]. Figure 5 summarizes the attractor benchmark results. The top panel shows the number of benchmarks completed (y-axis) until a specific time limit (x-axis). The bottom panel compares the performance on individual benchmarks. In Supplementary Text S6, we also provide an extended version of Figure 5, stratified across individual tools and network ensembles.

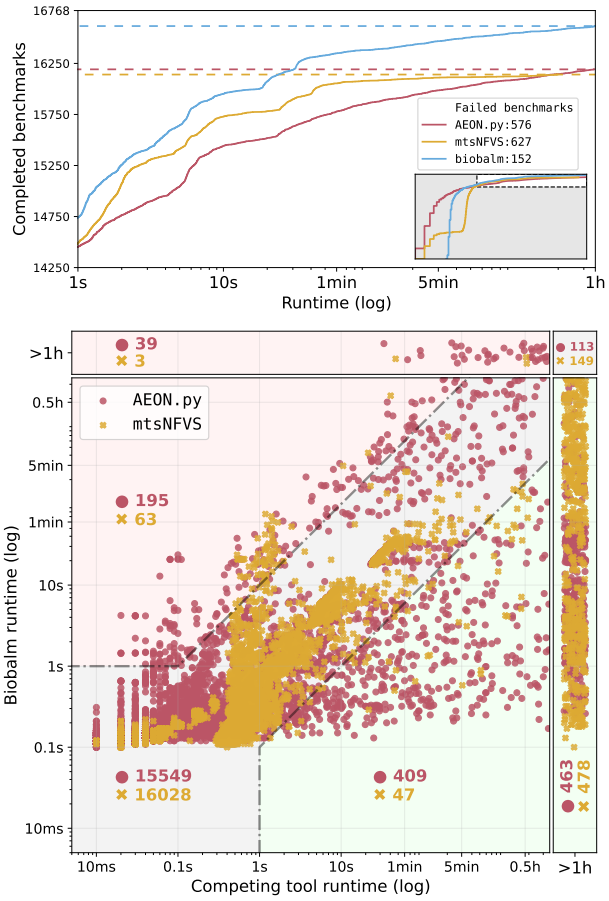


Figure 5: Performance comparison of `biobalm` versus `AEON.py` and `mts-nfvs` for attractor identification. Top plot shows the number of completed benchmarks before the 1h timeout. Note that the axes are cut-off to only cover models requiring more than one second to complete. The full range is shown as a miniature in the bottom right (and as a full figure in Supplementary Text S6). Bottom figure compares the runtime of individual benchmark instances. Red regions represent results where `biobalm` took longer than one second and was at least 10× slower than the competing tool, while green regions represent results where `biobalm` was at least 10× faster and the competing tool took longer than one second to complete. Edges of the plot show instances where a tool failed to complete the benchmark. Note that in both plots, the time axes are logarithmic.

Importantly, we have not encountered any motif-avoidant attractor in either of ensembles.

We have not included `pystablemotifs` in the attractor identification benchmarks because [48] demonstrated that `AEON.py` is superior for this task. However, we have tested `pystablemotifs` against `biobalm` on a smaller set of real-world models to evaluate succession diagram expansion and control. Results are provided in Supplementary Figure S8 and S9. In these tests, `biobalm` completed almost all benchmarks at least 10× faster than `pystablemotifs`.

4.2 Attractor landscape ensembles

To demonstrate the utility of `biobalm`, we have compared the full succession diagram structure of 230 ABN models of cell processes from the BBM dataset [56], in 14,010 parameter configurations, to the succession diagrams 69,000 random ABNs drawn from three null model ensembles. The null model ensembles include two ensembles of critical N-K models [10] with in-degree K=2 and K=3, and an ensemble of nested canalizing function (NCF) networks generated using the methods of [43]. The NCF networks have nested canalizing regulatory functions–meaning inputs determine outputs in a hierarchical manner–and topology matched to biological networks as reported by [57]. We generated 100 random networks with equal size matched to each BBM model, thereby resulting in three ensembles of 23,000 networks each. As shown in Figure 6, the distribution of succession diagram sizes for empirical models is more heterogeneous and has a longer tail than for our three ensembles of random networks. The mean, variance, and kurtosis of the BBM distribution are statistically significantly higher than for random networks (via bootstrapped 95% confidence intervals–see Supplementary Text S7.3). We observe a similar pattern in the number of attractors (see Supplementary Table S2). Additionally, we controlled for any automatically-generated models within BBM and observed similar results regardless of the model origin (Supplementary Text S7.4).

We observe that succession diagram size and the number of attractors grow exponentially with succes-

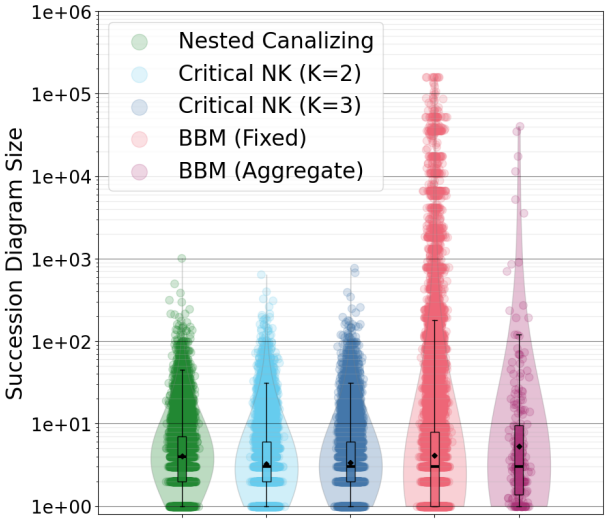


Figure 6: Distributions of succession diagram size (number of nodes in the fully-expanded succession diagram) for various ABN ensembles. BBM networks with randomly selected input configurations (up to 128 independent samples) are shown in red, with the average succession diagram size across these samples for each network depicted in purple. Random networks of different types are indicated in green, cyan, and dark blue, and are constructed to match the distribution of the number of variables in the BBM ensemble. Gaussian noise is added to the horizontal position of each point. Box plots and density plots are computed in log space.

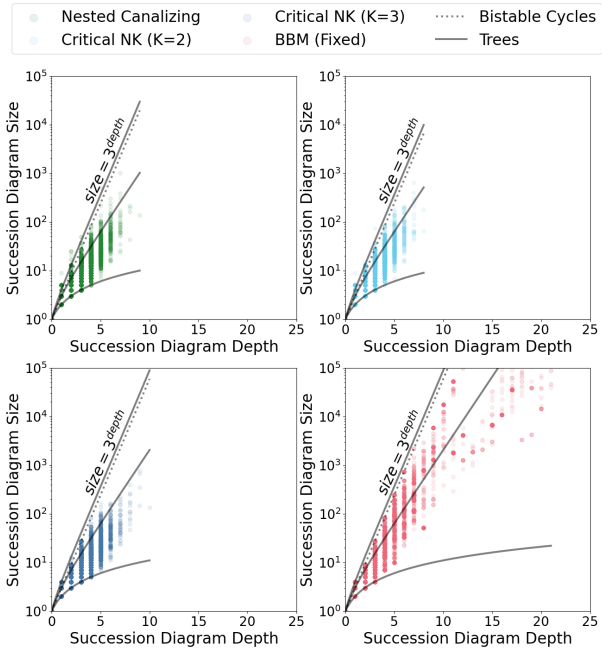


Figure 7: Succession diagram size scales exponentially as succession diagram depth increases. Reference lines are given corresponding to the scaling for trees with 1, 2, or 3 children per node (solid). Additionally, the scaling for networks consisting of independent bistable cycles is shown with a dotted line.

sion diagram depth (denoted d) in all four ensembles (Figure 7; Supplementary Figure S12). Note that the succession diagram is free to have any depth, any size, and any number of attractors. Succession diagram size is roughly bounded by 3^d, and the number of attractors is roughly bounded by 2^d, scaling bounds that correspond to networks of independent bistable cycles. Moreover, SD size as a function of depth is approximately bounded between the scaling obtained for uniform trees with three children and one child (linear chains), with many SDs falling near the scaling law obtained for uniform binary trees (two decisions for each SD node). These observations suggest that attractor landscapes are typically not broad and shallow–instead, attractor commitment arises from a series of decisions between a small number of possible choices.

5 Discussions

To extract biological insights from a Boolean network, one requires a strong understanding of the range of behaviors available to said network, as well as circumstances under which these behaviors arise. Despite recent major advances, the problem of identifying all the attractors in a large, dense ABN remains difficult. This problem is increasingly important as modelers of cell processes develop ever larger and more complete networks (and integrate them into population-level models). We have previously contributed to this problem individually [29, 48, 30], and here, we present a combined and significantly improved approach in the Python 3 library `biobalm`, the biologist's Boolean attractor landscape mapper.

The attractor identification benchmarks we have presented demonstrate a substantial improvement over `AEON.py` [48] (which has been previously demonstrated to significantly outperform `pystablemotifs` [29]) and a mild speed improvement over `mts-nfvs` [30]. However, more importantly, `biobalm` successfully analyzed 463 networks and 478 networks where `AEON.py` and `mts-nfvs` failed, respectively. Out of these, 75 networks were uniquely solved by `biobalm` (27 biological and 48 random). With a more generous timeout of two days, `biobalm`

only fails 26 benchmarks, primarily in complex models with more than 100.000 attractors that are limited by the time needed to enumerate all attractors. Overall, while `biobalm` is not the fastest tool on every benchmark, its performance is clearly the most robust among the tested tools. Crucially, `biobalm` is simultaneously computing the succession diagram, and therefore yields a much more informative output that describes the decision points in the network circuitry and can be used to perform computationally efficient attractor control using the control algorithms introduced in [29]. Compared to `pystablemotifs`, this results in a significant (10× or better) speed-up in control strategy identification.

Importantly, `biobalm` provides a modular approach to both succession diagram construction and attractor identification, enabling multiple methods and heuristics informed by the succession diagram (Figure 4). This allows to easily replace components (NFVS computation, minimal trap space computation, symbolic reachability, etc.) or introduce new methods as improvements become available.

In addition to benchmarks, we have used `biobalm` to study the distribution of succession diagram sizes in biological models from the Biodivine Boolean Models (BBM) repository [56], the largest curated collection of biological Boolean networks currently available. We compared these models to similarly sized critical Kauffman networks [10] (K=2 and K=3) as well as to an ensemble of canalizing random networks generated using the method of [44]. We observe that succession diagram depth scaling is consistent with attractor landscapes constructed from a series of small decisions, lending support to the hypothesis that biomolecular networks exhibit modular, hierarchal structure [58, 16]. We also find that the distribution of succession diagram size for published Boolean network models with fixed inputs is highly heterogeneous and contains extremely large succession diagrams (several thousand nodes), as compared to the succession diagram distributions for three biologically-inspired random model ensembles. Allowing inputs to vary in the published models would even further accentuate these differences. This suggests that, even accounting for network size, degree distribution, and the prevalence of canalizing functions in biological Boolean networks [57], they exhibit more complicated Waddington canalization landscapes at the system-level. Because the trap spaces that compose the succession diagram arise from positive feedback loops [21], we propose that non-local topological features of the influence graph (such as prevalence and overlap of cycles) play a significant role in the emergence of complex Waddington landscapes. It seems likely that local features (such as various measures of canalization in individual update functions [59]) are not sufficient to explain or predict the complexity of the attractor landscape—though they may still be informative in computing non-local metrics, e.g., via the effective graph [60]. Testing our hypothesis requires a deeper analysis of this phenomenon and remains as future work.

Ackowledgements

We thank Réka Albert for her helpful advice regarding the development of a new SD expansion strategy. We thank Luis M. Rocha for insightful discussions regarding the role of canalization in the results presented here.

Funding

Van-Giang Trinh was supported by Institut Carnot STAR, Marseille, France. Kyu Hyong Park was supported by NSF grant MCB1715826 to Réka Albert. Samuel Pastva has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Grant Agreement No. 101034413. Jordan C Rozum was supported by internal departmental funds provided by Luis M. Rocha. No funding bodies had any role in study design, analysis, decision to publish, or preparation of the manuscript.

Data availability

The tool `biobalm` is available online at `https://github.com/jcrozum/biobalm`. Further data,

scripts for testing, analysis and figure generation are available online at `https://github.com/jcrozum/biobalm-analysis` and in the reproducibility artefact at `https://doi.org/10.5281/zenodo.13854760`. The Supplementary Text is available online with the main paper.

Conflicts

The authors declare no conflicts of interest.

References

[1] George von Dassow, Eli Meir, Edwin M. Munro, and Garrett M. Odell. The segment polarity network is a robust developmental module. *Nat.*, 406(6792):188–192, July 2000.

[2] Dominik M Wittmann, Jan Krumsiek, Julio Saez-Rodriguez, Douglas A Lauffenburger, Steffen Klamt, and Fabian J Theis. Transforming Boolean models to continuous models: methodology and application to T-cell receptor signaling. *BMC Syst. Biol.*, 3(1):98, 2009.

[3] Jordan C. Rozum and Réka Albert. Controlling the cell cycle restriction switch across the information gradient. *Adv. Complex Syst.*, 22(07n08):1950020, November 2019. Publisher: World Scientific Publishing Co.

[4] L Mendoza, D Thieffry, and E R Alvarez-Buylla. Genetic control of flower morphogenesis in Arabidopsis thaliana: A logical analysis. *Bioinf.*, 15(7):593–606, July 1999.

[5] Réka Albert and Hans G Othmer. The topology of the regulatory interactions predicts the expression pattern of the segment polarity genes in Drosophila melanogaster. *J. Theor. Biol.*, 223(1):1–18, July 2003.

[6] Carlos Gershenson. Introduction to random Boolean networks. In *Proc. of ALife*, page 160–173. MIT Press, 2004.

[7] Rui-Sheng Wang, Assieh Saadatpour, and Reka Albert. Boolean modeling in systems biology: an overview of methodology and applications. *Phys. Biol.*, 9(5):055001, 2012.

[8] Julian D. Schwab, Silke D. Kühlwein, Nensi Ikonomi, Michael Kühl, and Hans A. Kestler. Concepts in Boolean network modeling: What do they all mean? *Comput. Struct. Biotechnol. J.*, 18:571–582, January 2020.

[9] Jordan C. Rozum, Colin Campbell, Eli Newby, Fatemeh Sadat Fatemi Nasrollahi, and Reka Albert. Boolean Networks as Predictive Models of Emergent Biological Behaviors, October 2023. arXiv:2310.12901 [nlin, q-bio].

[10] S. A. Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *J. Theor. Biol.*, 22(3):437–467, March 1969.

[11] Bernard Derrida and Yves Pomeau. Random networks of automata: a simple annealed approximation. *Europhys. Lett.*, 1(2):45, 1986.

[12] Bernard Derrida and Dietrich Stauffer. Phase transitions in two-dimensional kauffman cellular automata. *Europhys. Lett.*, 2(10):739, 1986.

[13] Bartolo Luque and Ricard V. Solé. Lyapunov exponents in random Boolean networks. *Physica A*, 284(1):33–45, September 2000.

[14] Enrique Balleza, Elena R. Alvarez-Buylla, Alvaro Chaos, Stuart Kauffman, Ilya Shmulevich, and Maximino Aldana. Critical Dynamics in Genetic Regulatory Networks: Examples from Four Kingdoms. *PLOS ONE*, 3(6):e2456, June 2008.

[15] Dávid Deritei, Jordan Rozum, Erzsébet Ravasz Regan, and Réka Albert. A feedback loop of conditionally stable circuits drives the cell cycle from checkpoint to checkpoint. *Sci. Rep.*, 9(1):16430, November 2019.

[16] Kyu Hyong Park, Felipe Xavier Costa, Luis M. Rocha, Réka Albert, and Jordan C. Rozum. Models of Cell Processes are Far from the Edge of Chaos. *PRX Life*, 1(2):023009, December 2023.

[17] Steven Nathaniel Steinway, Jorge G. T. Zañudo, Wei Ding, Carl Bart Rountree, David J. Feith, Thomas P. Loughran, and Reka Albert. Network Modeling of TGFβ Signaling in Hepatocellular Carcinoma Epithelial-to-Mesenchymal Transition Reveals Joint Sonic Hedgehog and Wnt Pathway Activation. *Cancer Res.*, 74(21):5963–5977, November 2014.

[18] Patrick Meyer, Pallab Maity, Andre Burkovski, Julian Schwab, Christoph Müssel, Karmveer Singh, Filipa F. Ferreira, Linda Krug, Harald J. Maier, Meinhard Wlaschek, Thomas Wirth, Hans A. Kestler, and Karin Scharffetter-Kochanek. A model of the onset of the senescence associated secretory phenotype after DNA damage induced senescence. *PLoS Comput. Biol.*, 13(12):e1005741, December 2017.

[19] Arnau Montagud, Jonas Béal, Luis Tobalina, Pauline Traynard, Vigneshwari Subramanian, Bence Szalai, Róbert Alföldi, László Puskás, Alfonso Valencia, Emmanuel Barillot, et al. Patient-specific Boolean models of signalling networks guide personalised treatments. *Elife*, 11:e72626, 2022.

[20] Evelina Folkesson, B Cristoffer Sakshaug, Andrea D Hoel, Geir Klinkenberg, and Åsmund Flobak. Synergistic effects of complex drug combinations in colorectal cancer cells predicted by logical modelling. *Front. Syst. Biol.*, 3:1112831, 2023.

[21] Jorge G. T. Zañudo and Réka Albert. An effective network reduction approach to find the dynamical repertoire of discrete dynamic networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 23(2), June 2013.

[22] Hannes Klarner, Alexander Bockmayr, and Heike Siebert. Computing maximal and minimal trap spaces of Boolean networks. *Nat. Comput.*, 14(4):535–544, 2015.

[23] Jordan C. Rozum and Réka Albert. Self-sustaining positive feedback loops in discrete and continuous systems. *J. Theor. Biol.*, 459:36–44, December 2018.

[24] Jordan C. Rozum and Réka Albert. Identifying (un)controllable dynamical behavior in complex networks. *PLoS Comput. Biol.*, 14(12):e1006630, December 2018.

[25] Conrad H Waddington. Canalization of development and the inheritance of acquired characters. *Nat.*, 150(3811):563–565, 1942.

[26] Loïc Paulevé, Juraj Kolčák, Thomas Chatain, and Stefan Haar. Reconciling qualitative, abstract, and scalable modeling of biological networks. *Nat. Commun.*, 11(1):1–7, August 2020.

[27] Aurélien Naldi, Adrien Richard, and Elisa Tonello. Linear cuts in Boolean networks. *Nat. Comput.*, pages 1–21, July 2023.

[28] 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.

[29] Jordan C Rozum, Dávid Deritei, Kyu Hyong Park, Jorge Gómez Tejeda Zañudo, and Réka Albert. pystablemotifs: Python library for attractor identification and control in Boolean networks. *Bioinform.*, 38(5):1465–1466, 2021.

[30] Van-Giang Trinh, Kunihiko Hiraishi, and Belaid Benhamou. Computing attractors of large-scale asynchronous Boolean networks using minimal trap spaces. In *Proc. of ACM-BCB*, pages 13:1–13:10. ACM, 2022.

[31] René Thomas. Boolean formalisation of genetic control circuits. *J. Theor. Biol.*, 42:565–583, 1973.

[32] René Thomas and Richard d'Ari. *Biological feedback*. CRC press, 1990.

[33] René Thomas. Regulatory networks seen as asynchronous automata: a logical description. *J. Theor. Biol.*, 153(1):1–23, 1991.

[34] El Houssine Snoussi and Rene Thomas. Logical identification of all steady states: The concept of feedback loop characteristic states. *Bull. Math. Biol.*, 55(5):973–991, 1993.

[35] D. Angeli, James E. Ferrell, Eduardo D. Sontag, and Joel L. Lebowitz. Detection of Multistability, Bifurcations, and Hysteresis in a Large Class of Biological Positive-Feedback Systems. *PNAS*, 101(7):1822–1827, 2004.

[36] D. Angeli and Eduardo D. Sontag. Multistability in monotone input/output systems. *Syst. Control Lett.*, 51(3-4):185–202, March 2004.

[37] Bernold Fiedler, Atsushi Mochizuki, Gen Kurosawa, and Daisuke Saito. Dynamics and Control at Feedback Vertex Sets. I: Informative and Determining Nodes in Regulatory Networks. *J. Dyn. Differ. Equations*, 25(3):563–604, September 2013.

[38] Atsushi Mochizuki, Bernold Fiedler, Gen Kurosawa, and Daisuke Saito. Dynamics and control at feedback vertex sets. II: A faithful monitor to determine the diversity of molecular activities in regulatory networks. *J. Theor. Biol.*, 335:130–146, October 2013.

[39] Jorge Gomez Tejeda Zañudo, Gang Yang, and Réka Albert. Structure-based control of complex networks with nonlinear dynamics. *PNAS*, 114(28):7234–7239, July 2017. Publisher: National Academy of Sciences Section: Physical Sciences.

[40] Trinh Van Giang, Tatsuya Akutsu, and Kunihiko Hiraishi. An FVS-based approach to attractor detection in asynchronous random Boolean networks. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 19(2):806–818, 2022.

[41] Lauren Marazzi, Milan Shah, Shreedula Balakrishnan, Ananya Patil, and Paola Vera-Licona. NETISCE: a network-based tool for cell fate reprogramming. *npj Syst. Biol. Appl.*, 8(1):21, June 2022.

[42] Jordan Rozum and Réka Albert. Leveraging network structure in nonlinear control. *npj Syst. Biol. Appl.*, 8(1):36, October 2022.

[43] Eli Newby, Jorge Gómez Tejeda Zañudo, and Réka Albert. Structure-based approach to identifying small sets of driver nodes in biological networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 32(6):063102, June 2022. Publisher: American Institute of Physics.

[44] Eli Newby, Jorge Gómez Tejeda Zañudo, and Réka Albert. Structure-based approach to identify driver nodes in ensembles of biologically inspired Boolean networks. *Phys. Rev. Res.*, 5:033009, Jul 2023.

[45] Tomoya Mori and Tatsuya Akutsu. Attractor detection and enumeration algorithms for Boolean networks. *Comput. Struct. Biotechnol. J.*, 20:2512–2520, 2022.

[46] Jorge G. T. Zañudo and Réka Albert. Cell fate reprogramming by control of intracellular network dynamics. *PLoS Comput. Biol.*, 11(4):e1004193, April 2015.

[47] Jordan C. Rozum, Jorge Gómez Tejeda Zañudo, Xiao Gan, Dávid Deritei, and Réka Albert. Parity and time reversal elucidate both decision-making in empirical models and attractor scaling in critical Boolean networks. *Sci. Adv.*, 7(29):eabf8124, July 2021.

[48] 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.

[49] Nikola Benes, Lubos Brim, Samuel Pastva, and David Safránek. Computing bottom SCCs symbolically using transition guided reduction. In *Proc. of CAV*, pages 505–528. Springer, 2021.

[50] Van-Giang Trinh, Belaid Benhamou, Kunihiko Hiraishi, and Sylvain Soliman. Minimal trap spaces of logical models are maximal siphons of

their Petri net encoding. In *Proc. of CMSB*, pages 158–176. Springer, 2022.

[51] Trinh Van Giang and Kunihiko Hiraishi. An improved method for finding attractors of large-scale asynchronous Boolean networks. In *Proc. of CIBCB*, pages 1–9. IEEE, 2021.

[52] Lucas Sánchez and Claudine Chaouiya. Primary sex determination of placental mammals: a modelling study uncovers dynamical developmental constraints in the formation of sertoli and granulosa cells. *BMC Syst. Biol.*, 10(1):1–11, 2016.

[53] Cui Su and Jun Pang. CABEAN 2.0: Efficient and efficacious control of asynchronous Boolean networks. In *Proc. of FM*, pages 581–598. Springer, 2021.

[54] Claus Kadelka, Reinhard Laubenbacher, David Murrugarra, Alan Veliz-Cuba, and Matthew Wheeler. Decomposition of Boolean networks: An approach to modularity of biological systems, June 2022.

[55] Loïc Paulevé. Pint: a static analyzer for transient dynamics of qualitative networks with ipython interface. In *Proc. of CMSB*, pages 309–316. Springer, 2017.

[56] Samuel Pastva, David Safranek, Nikola Benes, Lubos Brim, and Thomas Henzinger. Repository of logically consistent real-world Boolean network models. *bioRxiv*, 2023.

[57] Claus Kadelka, Taras-Michael Butrie, Evan Hilton, Jack Kinseth, Addison Schmidt, and Haris Serdarevic. A meta-analysis of Boolean network models reveals design principles of gene regulatory networks. *Sci. Adv.*, 10(2):eadj0822, 2024.

[58] Marta Sales-Pardo, Roger Guimerà, André A. Moreira, and Luís A. Nunes Amaral. Extracting the hierarchical organization of complex systems. *Proceedings of the National Academy of Sciences*, 104(39):15224–15229, September 2007.

[59] Rion B. Correia, Alexander J. Gates, Xuan Wang, and Luis M. Rocha. CANA: A Python Package for Quantifying Control and Canalization in Boolean Networks. *Front. Physiol.*, 9, 2018.

[60] Alexander J. Gates, Rion Brattig Correia, Xuan Wang, and Luis M. Rocha. The effective graph reveals redundancy, canalization, and control pathways in biochemical regulation and signaling. *PNAS*, 118(12):e2022598118, March 2021.